Hierarchical adaptive sparse grids for option pricing under the rough Bergomi model

Christian Bayer* Chiheb Ben Hammouda† Raul Tempone‡§

May 3, 2019

Abstract

The rough Bergomi (rBergomi) model, introduced recently in [4], is a promising rough volatility model in quantitative finance. This new model exhibits consistent results with the empirical fact of implied volatility surfaces being essentially time-invariant. This model also has the ability to capture the term structure of skew observed in equity markets. In the absence of analytical European option pricing methods for the model, and due to the non-Markovian nature of the fractional driver, the prevalent option is to use Monte Carlo (MC) simulation for pricing. Despite recent advances in the MC method in this context, pricing under the rBergomi model is still a time-consuming task. To overcome this issue, we design a novel, alternative, hierarchical approach, based on adaptive sparse grids quadrature (ASGQ), specifically using the same construction in [22], coupled with Brownian bridge construction and Richardson extrapolation. By uncovering the available regularity, our hierarchical method demonstrates substantial computational gains with respect to the standard MC method, when reaching a sufficiently small error tolerance in the price estimates across different parameter constellations, even for very small values of the Hurst parameter. Our work opens a new research direction in this field, i.e. to investigate the performance of methods other than Monte Carlo for pricing and calibrating under the rBergomi model.

Keywords Rough volatility, Monte Carlo, Adaptive sparse grids, Brownian bridge construction, Richardson extrapolation.

2010 Mathematics Subject Classification 91G60, 91G20, 65C05, 65D30, 65D32.

1 Introduction

Modeling the volatility to be stochastic, rather than deterministic as in the Black-Scholes model, enables quantitative analysts to explain certain phenomena observed in option price data, in particular the implied volatility smile. However, this family of models has a main drawback in failing to capture the true steepness of the implied volatility smile close to maturity. Jumps can be added

*Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Berlin, Germany.

†King Abdullah University of Science and Technology (KAUST), Computer, Electrical and Mathematical Sciences & Engineering Division (CEMSE), Thuwal 23955 − 6900, Saudi Arabia (chiheb.benhammouda@kaust.edu.sa).

‡King Abdullah University of Science and Technology (KAUST), Computer, Electrical and Mathematical Sciences & Engineering Division (CEMSE), Thuwal 23955 − 6900, Saudi Arabia (raul.tempone@kaust.edu.sa).

§Alexander von Humboldt Professor in Mathematics for Uncertainty Quantification, RWTH Aachen University, Germany.

to stock price models to overcome this undesired feature, for instance by modeling the stock price process as an exponential Lévy process. However, the addition of jumps to stock price processes remains controversial [14, 3].

Motivated by the statistical analysis of realized volatility by Gatheral, Jaisson and Rosenbaum [20] and the theoretical results on implied volatility [2, 18], rough stochastic volatility has emerged as a new paradigm in quantitative finance, overcoming the observed limitations of diffusive stochastic volatility models. In these models, the trajectories of the volatility have lower Hölder regularity than the trajectories of standard Brownian motion [4, 20]. In fact, they are based on fractional Brownian motion (fBm), which is a centered Gaussian process, whose covariance structure depends on the so-called Hurst parameter, H (we refer to [26, 16, 11] for more details regarding the fBm processes). In the rough volatility case, where $0 < H < 1/2$, the fBm has negatively correlated increments and rough sample paths. Gatheral, Jaisson, and Rosenbaum [20] empirically demonstrate the advantages of such models. For instance, they show that the log-volatility in practice has a similar behavior to fBm with the Hurst exponent $H \approx 0.1$ at any reasonable time scale (see also [19]). These results were confirmed by Bennedsen, Lunde and Pakkanen [8], who studied over a thousand individual US equities and showed that H lies in $(0, 1/2)$ for each equity. Other works [8, 4, 20] showed further benefits of such rough volatility models over standard stochastic volatility models, in terms of explaining crucial phenomena observed in financial markets.

One of the first rough volatility models is the rough Bergomi (rBergomi) model, developed by Bayer, Friz and Gatheral [4]. This model showed consistent behavior with the stylized fact of implied volatility surfaces being essentially time-invariant. It was also observed that this model is able to capture the term structure of skew observed in equity markets. The construction of the rBergomi model was performed by moving from a physical to a pricing measure and by simulating prices under that model to fit the implied volatility surface well in the case of the S&P 500 index with few parameters. The model may be seen as a non-Markovian extension of the Bergomi variance curve model [10].

Despite the promising features of the rBergomi model, pricing and hedging under such a model still constitutes a challenging and time-consuming task due to the non-Markovian nature of the fractional driver. In fact, the standard numerical pricing methods, such as: PDE discretization schemes, asymptotic expansions and transform methods, although efficient in the case of diffusion, are not easily carried over to the rough setting. Furthermore, due to the lack of Markovianity and affine structure, conventional analytical pricing methods do not apply. To the best of our knowledge, the only prevalent method for pricing options under such models is Monte Carlo (MC) simulation. In particular, recent advances in simulation methods for the rBergomi model and different variants of pricing methods based on MC under such a model have been proposed in [4, 5, 9, 28, 24]. For instance, in [28], the authors employ a novel composition of variance reduction methods. When pricing under the rBergomi model, they achieved substantial computational gains over the standard MC method. Greater analytical understanding of option pricing and implied volatility under this model has been achieved in [25, 17, 6]. It is crucial to note that hierarchical variance reduction methods, such as Multi-level Monte Carlo (MLMC), are inefficient in this context, because of the poor behavior of the strong error, that is of the order of H [31].

Despite the recent advances in the MC method, pricing under the rBergomi model is still a time-consuming task. To overcome this issue, we design a novel fast option pricer in this work, based on a hierarchical adaptive sparse grids quadrature (ASGQ), specifically using the same construction in [22], coupled with Brownian bridge construction and Richardson extrapolation, for options whose

underlyings follow the rBergomi model. To use ASGQ for our purposes, we solve two main issues that constitute the two stages of our newly designed method. In the first stage, we smoothen the integrand by using the conditional expectation as was proposed in [34], in the context of Markovian stochastic volatility models, and in [7], in the context of basket options. In a second stage, we apply ASGQ, to solve the integration problem. In this stage, we apply two transformations before using the ASGQ method, to overcome the issue of facing a high-dimensional integrand due to the discretization scheme used for simulating the rBergomi dynamics. Given that ASGQ benefits from anisotropy, the first transformation consists of applying a hierarchical path generation method, based on Brownian bridge (Bb) construction, with the aim of reducing the effective dimension. The second transformation consists of applying Richardson extrapolation to reduce the bias, which in turn reduces the needed number of time steps in the coarsest level to achieve a certain error tolerance and consequently the maximum number of dimensions needed for the integration problem. We emphasize that we are interested in the pre-asymptotic regime (corresponding to a small number of time steps), and the use of Richardson extrapolation is justified by our observed experimental results in that regime, which suggest, in particular, that we have convergence of order one for the weak error and that the pre-asymptotic regime is enough to get sufficiently accurate estimates for the option prices. Furthermore, we emphasize that no proper weak error analysis has been done in the rough volatility context.

Our first contribution is that we design a novel alternative approach based on ASGQ, in contrast to the aforementioned studies such as [28]. Given that the only prevalent option in this context is to use different variants of the MC method, our work opens a new research direction in this field, i.e. to investigate the performance of methods other than MC for pricing and calibrating under the rBergomi model. Our second contribution is that we reduce the computational cost through bias reduction by using Richardson extrapolation. Finally, assuming one targets price estimates with a sufficiently small error tolerance, our proposed method demonstrates substantial computational gains over the standard MC method, even for very small values of H. We show these gains through our numerical experiments for different parameter constellations. However, we do not claim that these gains will hold in the asymptotic regime, which requires higher accuracy. Furthermore, in this work, we limit ourselves to comparing our novel proposed method against the standard MC. A more systematic comparison with the variant of MC proposed in [28] can be carried out in future but has not been included in this work. Another potential direction of future research may also investigate the performance of quasi-Monte Carlo (QMC) for such problems.

The outline of this paper is as follows: We start in Section 2 by introducing the pricing framework that we are considering in this study. We provide some details about the rBergomi model, option pricing under this model and the simulation schemes used to simulate asset prices following the rBergomi dynamics. In Section 3.1, we provide a short weak error analysis in the context of the rBergomi and explain how we choose the optimal simulation scheme for an optimal performance of our approach. Then, in Section 4, we explain the different building blocks that constitute our proposed method, which are basically ASGQ, Brownian bridge construction, and Richardson extrapolation. Finally, in Section 5, we show the results obtained through the different numerical experiments conducted across different parameter constellations for the rBergomi model. The reported results show the high potential of our proposed method in this context.

2 Problem setting

In this section, we introduce the pricing framework that we are considering in this work. We start by giving some details for the rBergomi model proposed in [4]. We then derive the formula of the price of a European call option under the rBergomi model in Section 2.2. Finally, we explain some details about the schemes that we use to simulate the dynamics of asset prices under the rBergomi model.

2.1 The rBergomi model

We consider the rBergomi model for the price process S_t as defined in [4], normalized to $r = 0$[1], which is defined by

$$\begin{aligned} dS_t &= \sqrt{v_t} S_t dZ_t, \\ v_t &= \xi_0(t) \exp\left(\eta \widetilde{W}_t^H - \frac{1}{2}\eta^2 t^{2H}\right), \end{aligned} \tag{2.1}$$

where the Hurst parameter $0 < H < 1$ and $\eta > 0$. We refer to v_t as the variance process, and $\xi_0(t) = \mathrm{E}[v_t]$ is the forward variance curve. Here, $\widetilde{W}^H$ is a certain Riemann-Liouville fBm process[2], defined by

$$\widetilde{W}_t^H = \int_0^t K^H(t,s) dW_s^1, \quad t \geq 0, \tag{2.2}$$

where the kernel $K^H : \mathbb{R}_+ \to \mathbb{R}_+$ is

$$K^H(t-s) = \sqrt{2H}(t-s)^{H-1/2}, \quad \forall\, 0 \leq s \leq t. \tag{2.3}$$

By construction, $\widetilde{W}^H$ is a centered, locally $(H-\epsilon)$- Hölder continuous, Gaussian process with $\mathrm{Var}\left[\widetilde{W}_t^H\right] = t^{2H}$, and a dependence structure defined by

$$\mathrm{E}\left[\widetilde{W}_u^H \widetilde{W}_v^H\right] = u^{2H} G\left(\frac{v}{u}\right), \quad v > u,$$

where for $x \geq 1$ and $\gamma = \frac{1}{2} - H$

$$G(x) = 2H \int_0^1 \frac{ds}{(1-s)^\gamma (x-s)^\gamma}. \tag{2.4}$$

In (2.1) and (2.2), W^1, Z denote two *correlated* standard Brownian motions with correlation $\rho \in]-1, 0]$, so that we can represent Z in terms of W^1 as

$$Z = \rho W^1 + \overline{\rho} W^\perp = \rho W^1 + \sqrt{1-\rho^2} W^\perp,$$

[1] r is the interest rate.

[2] The so-called Riemann-Liouville processes are deduced from the standard Brownian motion by applying Riemann-Liouville fractional operators, whereas the standard fBm requires a weighted fractional operator [27, 33].

where $(W^1, W^\perp)$ are two independent standard Brownian motions. Therefore, the solution to (2.1), with $S(0) = S_0$, can be written as

$$
\begin{aligned}
S_t &= S_0 \exp\left(\int_0^t \sqrt{v(s)} dZ(s) - \frac{1}{2}\int_0^t v(s) ds\right), \quad S_0 > 0 \\
v_u &= \xi_0(u) \exp\left(\eta \widetilde{W}_u^H - \frac{\eta^2}{2} u^{2H}\right), \quad \xi_0 > 0.
\end{aligned} \tag{2.5}
$$

The filtration $(\mathcal{F}_t)_{t\geq 0}$ can here be taken as the one generated by the two-dimensional Brownian motion $(W^1, W^\perp)$ under the risk neutral measure $\mathbb{Q}$, resulting in a filtered probability space $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{Q})$. The stock price process S is clearly then a local $(\mathcal{F}_t)_{t\geq 0}$-martingale and a supermartingale. We shall henceforth use the notation $\mathrm{E}[.] = E^{\mathbb{Q}}[. \mid \mathcal{F}_0]$ unless we state otherwise.

Remark 2.1. The rBergomi model is non-Markovian in the instantaneous variance v_t, that is $E^{\mathbb{Q}}[v_u \mid \mathcal{F}_t] \neq= E^{\mathbb{Q}}[v_u \mid v_t]$. However, it is Markovian in the state vector by definition, that is $E^{\mathbb{Q}}[v_u \mid \mathcal{F}_t] = \xi_t(u)$.

2.2 Option pricing under the rBergomi model

We are interested in pricing European call options under the rBergomi model. Assuming $S_0 = 1$, and using the conditioning argument on the σ-algebra generated by W^1 (an argument first used by [34] in the context of Markovian stochastic volatility models), we can show that the call price is given by

$$
\begin{aligned}
C_{\mathrm{RB}}(T, K) &= \mathrm{E}\left[(S_T - K)^+\right] \\
&= \mathrm{E}\left[\mathrm{E}\left[(S_T - K)^+ \mid \sigma(W^1(t), t \leq T)\right]\right] \\
&= \mathrm{E}\left[C_{\mathrm{BS}}\left(S_0 = \exp\left(\rho \int_0^T \sqrt{v_t} dW_t^1 - \frac{1}{2}\rho^2 \int_0^T v_t dt\right),\ k = K,\ \sigma^2 = (1-\rho^2)\int_0^T v_t dt\right)\right],
\end{aligned} \tag{2.6}
$$

where $C_{\mathrm{BS}}(S_0, k, \sigma^2)$ denotes the Black-Scholes call price, for initial spot price S_0, strike price k and volatility σ^2.

We point out that the analytical smoothing, based on conditioning, performed in (2.6) enables us to uncover the available regularity, and hence get a smooth, analytic integrand inside the expectation. Therefore, applying sparse quadrature techniques becomes an adequate option for computing the call price as we will investigate later. A similar conditioning was used in [28] but for variance reduction purposes only.

2.3 Simulation of the rBergomi model

One of the numerical challenges encountered in the simulation of the rBergomi dynamics is the computation of $\int_0^T \sqrt{v_t} dW_t^1$ and $V = \int_0^T v_t dt$ in (2.6), mainly because of the singularity of the Volterra kernel $K^H(s,t)$ at the diagonal $s = t$. In fact, one needs to jointly simulate two Gaussian processes $(W_t^1, \widetilde{W}_t^H : 0 \leq t \leq T)$, resulting in $W_{t_1}^1, \ldots, W_{t_N}^1$ and $\widetilde{W}_{t_1}^H, \ldots, \widetilde{W}_{t_N}^H$ along a given time grid $t_1 < \cdots < t_N$. In the literature, there are essentially two suggested ways to achieve this:

i) **Covariance based approach (exact simulation) [4, 6]**: $W^1_{t_1}, \dots, W^1_{t_N}, \widetilde{W}^H_{t_1}, \dots, \widetilde{W}_{t_N}$ together form a $(2N)$-dimensional Gaussian random vector with computable covariance matrix, and therefore one can use Cholesky decomposition of the covariance matrix to produce exact samples of $W^1_{t_1}, \dots, W^1_{t_N}, \widetilde{W}^H_{t_1}, \dots, \widetilde{W}_{t_N}$ from $2N$-dimensional Gaussian random vector as input. This method is exact but slow. The simulation requires $\mathcal{O}\left(N^2\right)$ flops. Note that the offline cost is $\mathcal{O}\left(N^3\right)$ flops.

ii) **The hybrid scheme of [9]**: This scheme uses a different approach, which is essentially based on Euler discretization but crucially improved by moment matching for the singular term in the left point rule. It is also inexact in the sense that samples produced here do not exactly have the distribution of $W^1_{t_1}, \dots, W^1_{t_N}, \widetilde{W}^H_{t_1}, \dots, \widetilde{W}_{t_N}$, however they are much more accurate than samples produced from simple Euler discretization, but much faster than method (i). As in method (i), in this case, we need a $2N$-dimensional Gaussian random input vector to produce one sample of $W^1_{t_1}, \dots, W^1_{t_N}, \widetilde{W}^H_{t_1}, \dots, \widetilde{W}_{t_N}$.

In this work, we mainly use the hybrid scheme which, on equidistant grid $\{0, \frac{1}{N}, \frac{2}{N}, \dots, \frac{NT}{N}\}$, is given by the following,

(2.7)
$$\widetilde{W}^H_{\frac{i}{N}} \approx \overline{W}^H_{\frac{i}{N}} = \sqrt{2H}\left(\sum_{k=1}^{\min(i,\kappa)} \int_{\frac{i}{N}-\frac{k}{N}}^{\frac{i}{N}-\frac{k}{N}+\frac{1}{N}} \left(\frac{i}{N}-s\right)^{H-1/2} dW^1_s + \sum_{k=\kappa+1}^{i} \left(\frac{b_k}{N}\right)^{H-1/2} \int_{\frac{i}{N}-\frac{k}{N}}^{\frac{i}{N}-\frac{k}{N}+\frac{1}{N}} dW^1_s\right),$$

which results for $\kappa = 1$ in (2.8).

$$\widetilde{W}^H_{\frac{i}{N}} \approx \overline{W}^H_{\frac{i}{N}} = \sqrt{2H}\left(W^2_i + \sum_{k=2}^{i} \left(\frac{b_k}{N}\right)^{H-\frac{1}{2}} \left(W^1_{\frac{i-(k-1)}{N}} - W^1_{\frac{i-k}{N}}\right)\right), \tag{2.8}$$

where N is the number of time steps and

$$b_k = \left(\frac{k^{H+\frac{1}{2}} - (k-1)^{H+\frac{1}{2}}}{H+\frac{1}{2}}\right)^{\frac{1}{H-\frac{1}{2}}}.$$

The sum in (2.8) requires the most computational effort in the simulation. Given that (2.8) can be seen as discrete convolution (see [9]), we employ the fast Fourier transform to evaluate it, which results in $\mathcal{O}\left(N \log N\right)$ floating point operations.

We note that the variates $\overline{W}^H_0, \overline{W}^H_1, \dots, \overline{W}^H_{\frac{[Nt]}{N}}$ are generated by sampling $[Nt]$ i.i.d draws from a $(\kappa+1)$-dimensional Gaussian distribution and computing a discrete convolution. We denote these pairs of Gaussian random variables from now on by $(\mathbf{W}^{(1)}, \mathbf{W}^{(2)})$.

3 Weak error discussion and the optimal simulation scheme for our approach

To the best of our knowledge, no proper weak error analysis has been done in the rough volatility context. However, we try in Section 3.1 to shortly discuss it in the context of the rBergomi and then in Section 3.2, we explain how we choose the optimal simulation scheme for the optimal performance of our approach.

3.1 Weak error analysis

In this work, we are interested in approximating $\mathrm{E}\left[g(X_T\right]$, where g is some smooth function and X is the asset price under the rBergomi dynamics such that $X_t = X_t(W_t^{(1)}, \widetilde{W}_t)$ where $W^{(1)}$ is standard Brownian motion and $\widetilde{W}_t$ is the fractional Brownian motion as given by (2.2). Then we can express the approximation of $\mathrm{E}\left[g(X_T\right]$ using the hybrid and exact schemes as the following

$$\mathrm{E}\left[g\left(X_T\left(W_t^{(1)}, \widetilde{W}_t\right)\right)\right] \approx \mathrm{E}\left[g\left(\overline{X}_N\left(W_1^{(1)}, \ldots, W_N^{(1)}, \overline{W}_1, \ldots, \overline{W}_N\right)\right)\right] : \quad \textbf{(Hybrid scheme)},$$

$$\mathrm{E}\left[g\left(X_T\left(W_t^{(1)}, \widetilde{W}_t\right)\right)\right] \approx \mathrm{E}\left[g\left(\overline{X}_N\left(W_1^{(1)}, \ldots, W_N^{(1)}, \widetilde{W}_1, \ldots, \widetilde{W}_N\right)\right)\right] : \quad \textbf{(Exact scheme)},$$

where $\overline{W}$ is the approximation of $\widetilde{W}$ as given by (2.7) and $\overline{X}_N$ is the approximation of X using N time steps.

To simplify notation, let $\overline{\mathbf{W}} = (\overline{W}_1, \ldots, \overline{W}_N)$, $\mathbf{W}^1 = (W_1^{(1)}, \ldots, W_N^{(1)})$ and $\widetilde{\mathbf{W}} = (\widetilde{W}_1, \ldots, \widetilde{W}_N)$. If we denote by $\mathcal{E}_B^{Hyb}$ and $\mathcal{E}_B^{Chol}$ the weak errors produced by the hybrid and Cholesky scheme respectively, then we can write

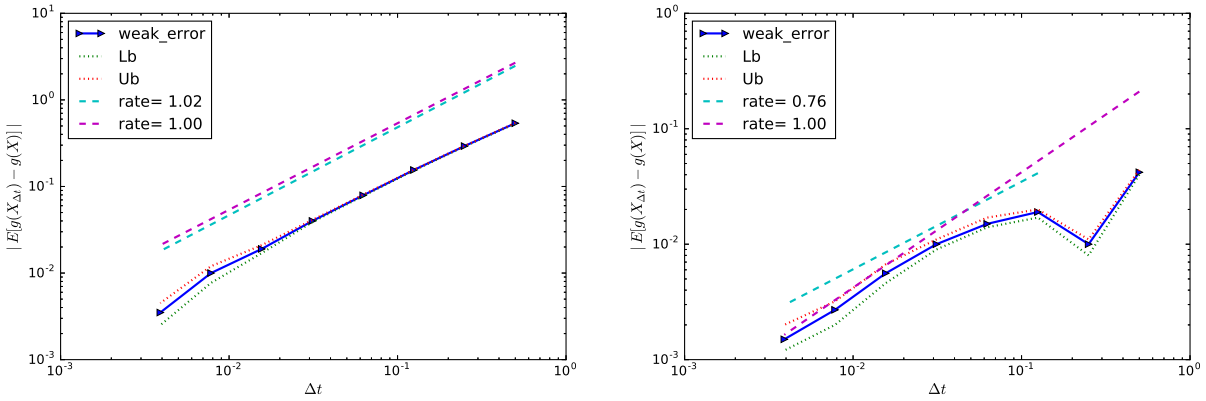
$$\begin{aligned}
\mathcal{E}_B^{Hyb} &= \left|\mathrm{E}\left[g\left(X_T\left(W_t^{(1)}, \widetilde{W}_t\right)\right)\right] - \mathrm{E}\left[g\left(\overline{X}_N\left(\mathbf{W}^1, \overline{\mathbf{W}}\right)\right)\right]\right| \\
&\leq \left|\mathrm{E}\left[g\left(X_T\left(W_t^{(1)}, \widetilde{W}_t\right)\right)\right] - \mathrm{E}\left[g\left(\overline{X}_N\left(\mathbf{W}^1, \widetilde{\mathbf{W}}\right)\right)\right]\right| + \left|\mathrm{E}\left[g\left(\overline{X}_N\left(\mathbf{W}^1, \overline{\mathbf{W}}\right)\right)\right] - \mathrm{E}\left[g\left(\overline{X}_N\left(\mathbf{W}^1, \widetilde{\mathbf{W}}\right)\right)\right]\right| \\
&\leq \mathcal{E}_B^{Chol} + \left|\mathrm{E}\left[g\left(\overline{X}_N\left(\mathbf{W}^1, \overline{\mathbf{W}}\right)\right)\right] - \mathrm{E}\left[g\left(\overline{X}_N\left(\mathbf{W}^1, \widetilde{\mathbf{W}}\right)\right)\right]\right|
\end{aligned} \tag{3.1}$$

From the construction of the Cholesky scheme, we expect that the weak error is purely the discretization error, that is

$$\mathcal{E}_B^{Chol} = \mathcal{O}\left(\Delta t\right),$$

which was confirmed by our numerical experiments (for illustration see Figure 3.1b for the case of Set 1 in Table 5.1).

The second term in (3.1) is basically related to approximating the integral (2.2) by (2.8). From our numerical experiments it seems that that term is also of order at least Δt and its rate of convergence is independent of H (for illustration see Figure 3.1a for the case of Set 1 in Table 5.1).



(a) (b)

Figure 3.1: The convergence of the weak error $\mathcal{E}_B$, using MC with 6×10^6 samples, for Set 1 parameter in Table 5.1. We refer to C_{RB} (as in (2.6)) for $\mathrm{E}\left[g(X)\right]$, and to C^N_{RB} (as in (4.1)) for $\mathrm{E}\left[g(X_{\Delta t})\right]$. The upper and lower bounds are 95% confidence intervals. a) With the hybrid scheme b) With the exact scheme.

3.2 On the choice of the simulation scheme in our approach

The choice of the simulation scheme in our approach was based on the observed behavior of the weak rates. Through our numerical experiments (see Table 5.1 for the tested examples), we observe that although the hybrid and exact schemes converge asymptotically with weak rate of $\mathcal{O}\left(\Delta t\right)$, the pre-asymptotic behavior of the weak rate is different for both schemes. As an illustration, from Figure 5.1 for Set 1 parameter in Table 5.1, the hybrid scheme has a consistent convergence behavior in the sense that it behaves in the asymptotic way basically right from the beginning, whereas the exact scheme does not. On the other hand, the constant seems considerably smaller for the exact scheme. These two features makes the hybrid scheme the optimal scheme to work with in our context since our approach is based on hierarchical transformations involving the use of Richardson extrapolation (see Section 4.4).

4 Details of our hierarchical method

We recall that our goal is to compute the expectation in (2.6). In fact, as seen in Section 2.3, we need $2N$-dimensional Gaussian inputs for the used hybrid scheme (N is the number of time steps in the time grid), namely

- $\mathbf{W}^{(1)} = \{W_i^{(1)}\}_{i=1}^N$: The N Gaussian random variables that are defined in Section 2.1.
- $\mathbf{W}^{(2)} = \{W_j^{(2)}\}_{j=1}^N$: An artificially introduced N Gaussian random variables that are used for left-rule points in the hybrid scheme, as explained in Section 2.3.

We can rewrite (2.6) as

$$
\begin{aligned}
C_{\mathrm{RB}}(T,K) &= \mathrm{E}\left[C_{\mathrm{BS}}\left(S_0 = \exp\left(\rho\int_0^T \sqrt{v_t}dW_t^1 - \frac{1}{2}\rho^2\int_0^T v_t dt\right),\ k=K,\ \sigma^2 = (1-\rho^2)\int_0^T v_t dt\right)\right] \\
&\approx \int_{\mathbb{R}^{2N}} C_{BS}\left(G(\mathbf{w}^{(1)},\mathbf{w}^{(2)})\right)\rho_N(\mathbf{w}^{(1)})\rho_N(\mathbf{w}^{(2)})d\mathbf{w}^{(1)}d\mathbf{w}^{(2)} \\
&:= C_{RB}^N, \qquad (4.1)
\end{aligned}
$$

where G maps $2N$ independent standard Gaussian random inputs to the parameters fed to Black-Scholes formula, and ρ_N is the multivariate Gaussian density, given by

$$\rho_N(\mathbf{z}) = \frac{1}{(2\pi)^{N/2}} e^{-\frac{1}{2}\mathbf{z}^T\mathbf{z}}.$$

Therefore, the initial integration problem that we are solving lives in $2N$-dimensional space, which becomes very large as the number of time steps N, used in the hybrid scheme, increases.

Our approach of approximating the expectation in (4.1) is based on hierarchical ASGQ, using the same construction in [22]. We describe the ASGQ method in our context in Section 4.1. To make an effective use of the ASGQ method, we first apply two transformations to overcome the issue of facing a high dimensional integrand due to the discretization scheme used for simulating the rBergomi dynamics. The first transformation consists of applying a hierarchical path generation method, based on Brownian bridge (Bb) construction, with the aim of reducing the effective dimension as described in Section 4.3. The second transformation consists of applying Richardson extrapolation to reduce the bias, resulting in reducing the maximum number of dimensions needed for the integration problem. Details about Richardson extrapolation are provided in Section 4.4.

If we denote by $\mathcal{E}_{\mathrm{tot}}$ the total error of approximating the expectation in (2.6) using the ASGQ estimator, Q_N, then we have a natural error decomposition

$$\mathcal{E}_{\mathrm{tot}} \leq \left|C_{\mathrm{RB}} - C_{\mathrm{RB}}^N\right| + \left|C_{\mathrm{RB}}^N - Q_N\right| \leq \mathcal{E}_B(N) + \mathcal{E}_Q(\mathrm{TOL}_{\mathrm{ASGQ}}, N), \qquad (4.2)$$

where $\mathcal{E}_Q$ is the quadrature error, $\mathcal{E}_B$ is the bias, $\mathrm{TOL}_{\mathrm{ASGQ}}$ is a user selected tolerance for ASGQ method, and C_{RB}^N is the biased price computed with N time steps as given by (4.1).

4.1 Adaptive sparse grids quadrature (ASGD)

We assume that we want to approximate the expected value $\mathrm{E}[f(Y)]$ of an analytic function $f\colon \Gamma \to \mathbb{R}$ using a tensorization of quadrature formulas over Γ.

To introduce simplified notations, we start with the one-dimensional case. Let us denote by β a non-negative integer, referred to as a "stochastic discretization level", and by $m : \mathbb{N} \to \mathbb{N}$ a strictly increasing function with $m(0) = 0$ and $m(1) = 1$, that we call "level-to-nodes function". At level β, we consider a set of $m(\beta)$ distinct quadrature points in $\mathbb{R}$, $\mathcal{H}^{m(\beta)} = \{y_\beta^1, y_\beta^2, \ldots, y_\beta^{m(\beta)}\} \subset \mathbb{R}$, and a set of quadrature weights, $\boldsymbol{\omega}^{m(\beta)} = \{\omega_\beta^1, \omega_\beta^2, \ldots, \omega_\beta^{m(\beta)}\}$. We also let $C^0(\mathbb{R})$ be the set of real-valued continuous functions over $\mathbb{R}$. We then define the quadrature operator as

$$Q^{m(\beta)} : C^0(\mathbb{R}) \to \mathbb{R}, \quad Q^{m(\beta)}[f] = \sum_{j=1}^{m(\beta)} f(y_\beta^j)\omega_\beta^j.$$

In our case, we have in (4.1) a multi-variate integration problem with, $f = C_{\mathrm{BS}} \circ G$, $\mathbf{Y} = (\mathbf{W}^{(1)}, \mathbf{W}^{(2)})$, and $\Gamma = \mathbb{R}^{2N}$, in the previous notations. Furthermore, since we are dealing with Gaussian densities, using Gauss-Hermite quadrature points is the appropriate choice.

We define for any multi-index $\boldsymbol{\beta} \in \mathbb{N}^{2N}$

$$Q^{m(\boldsymbol{\beta})} : C^0(\mathbb{R}^{2N}) \to \mathbb{R}, \quad Q^{m(\boldsymbol{\beta})} = \bigotimes_{n=1}^{2N} Q^{m(\beta_n)},$$

where the n-th quadrature operator is understood to act only on the n-th variable of f. Practically, we obtain the value of $Q^{m(\boldsymbol{\beta})}[f]$ by using the grid $\mathcal{T}^{m(\boldsymbol{\beta})} = \prod_{n=1}^{2N} \mathcal{H}^{m(\beta_n)}$, with cardinality $\#\mathcal{T}^{m(\boldsymbol{\beta})} = \prod_{n=1}^{2N} m(\beta_n)$, and computing

$$Q^{m(\boldsymbol{\beta})}[f] = \sum_{j=1}^{\#\mathcal{T}^{m(\boldsymbol{\beta})}} f(\widehat{y}_j)\overline{\omega}_j,$$

where $\widehat{y}_j \in \mathcal{T}^{m(\boldsymbol{\beta})}$ and $\overline{\omega}_j$ are products of weights of the univariate quadrature rules. To simplify notation, hereafter, we replace $Q^{m(\boldsymbol{\beta})}$ by $Q^{\boldsymbol{\beta}}$.

A direct approximation $\mathrm{E}\,[f[\mathbf{Y}]] \approx Q^{\boldsymbol{\beta}}[f]$ is not an appropriate option due to the well-known "curse of dimensionality". We use a hierarchical ASGQ[3] strategy, specifically using the same construction as in [22], and which uses stochastic discretizations and a classic sparsification approach to obtain an effective approximation scheme for $\mathrm{E}\,[f]$.

To be concrete, in our setting, we are left with a $2N$-dimensional Gaussian random input, which is chosen independently, resulting in $2N$ numerical parameters for ASGQ, which we use as the basis of the multi-index construction. For a multi-index $\boldsymbol{\beta} = (\beta_n)_{n=1}^{2N} \in \mathbb{N}^{2N}$, we denote by $Q_N^{\boldsymbol{\beta}}$, the result of approximating (4.1) with a number of quadrature points in the i-th dimension equal to $m(\beta_i)$. We further define the set of differences $\Delta Q_N^{\boldsymbol{\beta}}$ as follows: for a single index $1 \le i \le 2N$, let

$$\Delta_i Q_N^{\boldsymbol{\beta}} = \begin{cases} Q_N^{\boldsymbol{\beta}} - Q_N^{\boldsymbol{\beta}'}, \text{ with } \boldsymbol{\beta}' = \boldsymbol{\beta} - e_i, \text{ if } \boldsymbol{\beta}_i > 0, \\ Q_N^{\boldsymbol{\beta}}, \quad \text{otherwise,} \end{cases}$$

where e_i denotes the ith $2N$-dimensional unit vector. Then, $\Delta Q_N^{\boldsymbol{\beta}}$ is defined as

$$\Delta Q_N^{\boldsymbol{\beta}} = \left(\prod_{i=1}^{2N} \Delta_i \right) Q_N^{\boldsymbol{\beta}}.$$

For instance, when $N = 1$, then

$$\begin{aligned} \Delta Q_1^{\boldsymbol{\beta}} = \Delta_2 \Delta_1 Q_1^{(\beta_1,\beta_2)} = \Delta_2 \left(Q_1^{(\beta_1,\beta_2)} - Q_1^{(\beta_1-1,\beta_2)} \right) &= \Delta_2 Q_1^{(\beta_1,\beta_2)} - \Delta_2 Q_1^{(\beta_1-1,\beta_2)} \\ &= Q_1^{(\beta_1,\beta_2)} - Q_1^{(\beta_1,\beta_2-1)} - Q_1^{(\beta_1-1,\beta_2)} + Q_1^{(\beta_1-1,\beta_2-1)}. \end{aligned}$$

Given the definition of C_{RB}^N by (4.1), we have the telescoping property

$$C_{RB}^N = Q_N^\infty = \sum_{\beta_1=0}^{\infty} \cdots \sum_{\beta_{2N}=0}^{\infty} \Delta Q_N^{(\beta_1,\ldots,\beta_{2N})} = \sum_{\boldsymbol{\beta} \in \mathbb{N}^{2N}} \Delta Q_N^{\boldsymbol{\beta}}.$$

[3] More details about sparse grids can be found in [12].

The ASGQ estimator used for approximating (4.1), and using a set of multi-indices $\mathcal{I} \subset \mathbb{N}^{2N}$ is given by

$$Q_N^{\mathcal{I}} = \sum_{\boldsymbol{\beta} \in \mathcal{I}} \Delta Q_N^{\boldsymbol{\beta}}. \tag{4.3}$$

The quadrature error in this case is given by

$$\mathcal{E}_Q(\mathrm{TOL}_{\mathrm{ASGQ}}, N) = \left| Q_N^{\infty} - Q_N^{\mathcal{I}} \right| \leq \sum_{\boldsymbol{\beta} \in \mathbb{N}^{2N} \setminus \mathcal{I}} \left| \Delta Q_N^{\boldsymbol{\beta}} \right|. \tag{4.4}$$

We define the work contribution, $\Delta \mathcal{W}_{\boldsymbol{\beta}}$, to be the computational cost required to add $\Delta Q_N^{\boldsymbol{\beta}}$ to $Q_N^{\mathcal{I}}$, and the error contribution, $\Delta E_{\boldsymbol{\beta}}$, to be a measure of how much the quadrature error, defined in (4.4), would decrease once $\Delta Q_N^{\boldsymbol{\beta}}$ has been added to $Q_N^{\mathcal{I}}$, that is

$$\begin{aligned} \Delta \mathcal{W}_{\boldsymbol{\beta}} &= \mathrm{Work}[Q_N^{\mathcal{I} \cup \{\boldsymbol{\beta}\}}] - \mathrm{Work}[Q_N^{\mathcal{I}}] \\ \Delta E_{\boldsymbol{\beta}} &= \left| Q_N^{\mathcal{I} \cup \{\boldsymbol{\beta}\}} - Q_N^{\mathcal{I}} \right|. \end{aligned} \tag{4.5}$$

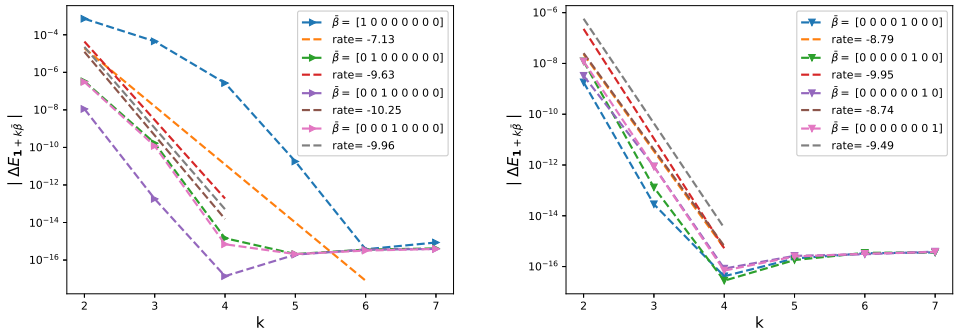
The construction of the optimal $\mathcal{I}$ will be done by profit thresholding, that is, for a certain threshold value $\overline{T}$, and a profit of a hierarchical surplus defined by

$$P_{\boldsymbol{\beta}} = \frac{|\Delta E_{\boldsymbol{\beta}}|}{\Delta \mathcal{W}_{\boldsymbol{\beta}}},$$

where the optimal index set $\mathcal{I}$ for our ASGQ is given by $\mathcal{I} = \{\boldsymbol{\beta} : P_{\boldsymbol{\beta}} \geq \overline{T}\}$.

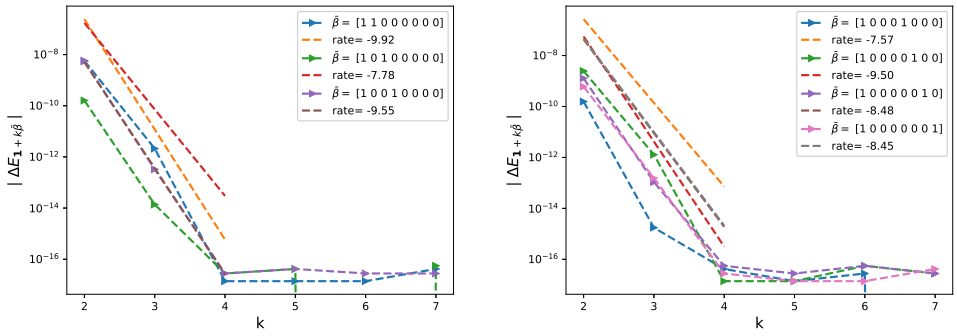
Remark 4.1. The choice of the hierarchy of quadrature points, $m(\boldsymbol{\beta})$, is flexible in the ASGQ algorithm and can be fixed by the user, depending on the convergence properties of the problem at hand. For instance, for the sake of reproducibility, in our numerical experiments we used a linear hierarchy: $m(\beta) = 4(\beta - 1) + 1$, $1 \leq \beta$, for results of parameter set 1 in Table 5.1. For the remaining parameter sets in Table 5.1, we used a geometric hierarchy: $m(\beta) = 2^{\beta - 1} + 1$, $1 \leq \beta$.

Remark 4.2. As emphasized in [22], one important requirement to get the optimal performance of the ASGQ is to check the error convergence, defined by (4.5), of first and mixed difference operators. We checked this requirement in all our numerical experiments, and for illustration, we show in Figures 4.1 and 4.2, the error convergence of first and second order differences for the case of parameter set 2 in Table 5.1. These plots show that: i) $\Delta \mathrm{E}_{\boldsymbol{\beta}}$ decreases exponentially fast with respect to β_i, and ii) $\Delta \mathrm{E}_{\boldsymbol{\beta}}$ has a product structure since we observe a faster error decay for second differences compared to corresponding first difference operators.



(a) (b)

Figure 4.1: The rate of error convergence of first order differences $|\Delta \mathrm{E}_{\boldsymbol{\beta}}|$, defined by (4.5), ($\boldsymbol{\beta} = \mathbf{1} + k\overline{\boldsymbol{\beta}}$) with respect to $\mathbf{W}^{(1)}$ (a) and with respect to $\mathbf{W}^{(2)}$ (b), for parameter set 2 in Table 5.1. The number of quadrature points used in the i-th dimension is $N_i = 2^{\beta_i - 1} + 1$.



(a) (b)

Figure 4.2: The rate of error convergence of second order differences $|\Delta \mathrm{E}_{\boldsymbol{\beta}}|$, defined by (4.5), ($\boldsymbol{\beta} = \mathbf{1} + k\overline{\boldsymbol{\beta}}$) with respect to $\mathbf{W}^{(1)}$ (a) and with respect to $\mathbf{W}^{(2)}$ (b), for parameter set 2 in Table 5.1. The number of quadrature points used in the i-th dimension is $N_i = 2^{\beta_i - 1} + 1$.

Remark 4.3. The analiticity assumption, stated in the beginning of Section 4.1, is crucial for the optimal performance of our proposed method. In fact, although we face the issue of the "curse of dimensionality" when increasing N, the analiticity of f implies a spectral convergence for sparse grids quadrature.

4.2 Quasi Monte Carlo (QMC)

In this work, we use QMC to compare with MC and ASGQ. Specifically, we use the lattice rules family of QMC [35, 15, 32]. The main input for the lattice rule is one integer vector with d component (d dimension of the integration problem).

In fact, given an integer vector $z = (z_1, \ldots, z_d)$ known as *the generating vector*, a (rank-1) lattice rule with n points takes the form

$$Q_n(f) := \frac{1}{n}\sum_{k=0}^{n-1} f\left(\frac{kz \bmod n}{n}\right). \tag{4.6}$$

The quality of the lattice rule depends on the choice of the generating vector. Due to the modulo operation, it suffices to consider the values from 1 up to $n-1$, leaving out 0 which is clearly a bad choice. Furthermore, we restrict the values to those relatively prime to n, to ensure that every one-dimensional projection of the n points yields n distinct values. Thus, we write $\mathbf{z} \in \mathbb{U}_n^d$, with $\mathbb{U}_n := \{z \in zset : 1 \le z \le n-1 \text{ and } \gcd(z,n) = 1\}$.

For theoretical analysis one often assumes that n is prime to simplify some number theory arguments. For practical application, n is often taken to be a power of 2. The total number of possible choices for the generating vector is then $(n-1)^d$ and $(n/2)^d$, respectively. Even if we have a criterion to assess the quality of the generating vectors, there are simply too many choices to carry out an exhaustive search when n and d are large.

To get an unbiased approximation of the integral, we use a randomly shifted lattice rule, which also allows us to obtain a practical error estimate in the same way as the MC method. It works as follows. We generate q independent random shifts $\Delta^{(i)}$ for $i = 0, \ldots, q-1$ from the uniform distribution on $[0,1]^d$. For the same fixed lattice generating vector z, we compute the q different shifted lattice rule approximations and denote them by $Q_n^{(i)}(f)$ for $i = 0, \ldots, q-1$.We then take the average

$$\overline{Q}_{n,q}(f) = \frac{1}{q}\sum_{i=0}^{q-1} Q_n^{(i)}(f) = \frac{1}{q}\sum_{i=0}^{q-1}\left(\frac{1}{n}\sum_{k=0}^{n-1} f\left(\frac{kz + \Delta^{(i)} \bmod n}{n}\right)\right) \tag{4.7}$$

as our final approximation to the integral.

We note that since we are dealing with Gaussian randomness and with integrals in infinite support, we use the inverse of the standard normal cumulative distribution function as a pre-transformation to map the problem to $[0,1]$ and then use QMC.

In our numerical test, we use a pre-made point generators using latticeseq_b2.py in python from `https://people.cs.kuleuven.be/~dirk.nuyens/qmc-generators/`.

4.3 Brownian bridge construction

In the literature of ASGQ and QMC, several hierarchical path generation methods (PGMs) or transformation methods have been proposed to reduce the effective dimension. Among these transformations, we cite the Brownian bridge (Bb) construction [29, 30, 13], the principal component analysis (PCA) [1] and the linear transformation (LT) [23].

In our context, the Brownian motion on a time discretization can be constructed either sequentially using a standard random walk construction, or hierarchically using other PGMs as listed above. For our purposes, to make an effective use of ASGQ , which benefits from anisotropy, we use the Bb construction since it produces dimensions with different importance, contrary to a random walk procedure for which all the dimensions of the stochastic space have equal importance. In fact, Bb uses the first several coordinates of the low-discrepancy points to determine the general shape of the Brownian path, and the last few coordinates influence only the fine detail of the path.

Consequently, this transformation reduces the effective dimension of the problem, which results in accelerating the ASGQ method by reducing the computational cost.

Let us denote $\{t_i\}_{i=0}^N$ the grid of time steps. Then the Bb construction [21] consists of the following: given a past value B_{t_i} and a future value B_{t_k}, the value B_{t_j} (with $t_i < t_j < t_k$) can be generated according to

$$B_{t_j} = (1-\rho)B_{t_i} + \rho B_{t_k} + \sqrt{\rho(1-\rho)(k-i)\Delta t}z,\ z \sim \mathcal{N}(0,1),$$

where $\rho = \frac{j-i}{k-i}$.

4.4 Richardson extrapolation

Another transformation that we couple with the ASGQ is Richardson extrapolation [36]. In fact, applying level K_{R} (level of extrapolation) of Richardson extrapolation dramatically reduces the bias, and as a consequence reduces the number of time steps N needed in the coarsest level to achieve a certain error tolerance. As a consequence, Richardson extrapolation directly reduces the total dimension of the integration problem for achieving some error tolerance.

Let us denote by $(X_t)_{0\le t\le T}$ a certain stochastic process and by $(\widehat{X}^h_{t_i})_{0\le t_i\le T}$ its approximation using a suitable scheme with a time step h. Then, for sufficiently small h, and a suitable smooth function f, we assume that

$$\mathrm{E}\left[f(\widehat{X}^h_T)\right] = \mathrm{E}\left[f(X_T)\right] + ch + \mathcal{O}\left(h^2\right). \tag{4.8}$$

Applying (4.8) with discretization step $2h$, we obtain

$$\mathrm{E}\left[f(\widehat{X}^{2h}_T)\right] = \mathrm{E}\left[f(X_T)\right] + 2ch + \mathcal{O}\left(h^2\right),$$

implying

$$2\mathrm{E}\left[f(\widehat{X}^{2h}_T)\right] - \mathrm{E}\left[f(\widehat{X}^h_T)\right] = \mathrm{E}\left[f(X_T)\right] + \mathcal{O}\left(h^2\right).$$

For higher levels of extrapolations, we use the following: Let us denote by $h_J = h_0 2^{-J}$ the grid sizes (where h_0 is the coarsest grid size), by K_{R} the level of the Richardson extrapolation, and by $I(J, K_{\mathrm{R}})$ the approximation of $\mathrm{E}\left[f((X_T)\right]$ by terms up to level K_{R} (leading to a weak error of order K_{R}), then we have the following recursion

$$I(J,K_{\mathrm{R}}) = \frac{2^{K_{\mathrm{R}}}\left[I(J,K_{\mathrm{R}}-1) - I(J-1,K_{\mathrm{R}}-1)\right]}{2^{K_{\mathrm{R}}}-1}, \quad J = 1,2,\ldots, K_{\mathrm{R}} = 1,2,\ldots$$

Remark 4.4. We emphasize that throughout our work, we are interested in the pre-asymptotic regime (a small number of time steps), and the use of Richardson extrapolation is justified by our observed experimental results in that regime (see Section 5.1), which suggest a convergence of order one for the weak error.

5 Numerical experiments

In this section, we show the results obtained through the different numerical experiments, conducted across different parameter constellations for the rBergomi model. Details about these examples are presented in Table 5.1. The first set is the one that is closest to the empirical findings [8, 20], which suggest that $H \approx 0.1$. The choice of parameters values of $\nu = 1.9$ and $\rho = -0.9$ is justified by [4], where it is shown that these values are remarkably consistent with the SPX market on 4th February 2010. For the remaining three sets in Table 5.1, we wanted to test the potential of our method for a very rough case, that is $H = 0.02$, for three different scenarios of moneyness, S_0/K. In fact, hierarchical variance reduction methods, such as Multi-level Monte Carlo (MLMC), are inefficient in this context, because of the poor behavior of the strong error, that is of the order of H [31]. We emphasize that we checked the robustness of our method for other parameter sets, but for illustrative purposes, we only show results for the parameters sets presented in Table 5.1. For all our numerical experiments, we consider a number of time steps $N \in \{2, 4, 8, 16\}$, and all reported errors are relative errors, normalized by the reference solutions provided in Table 5.1.

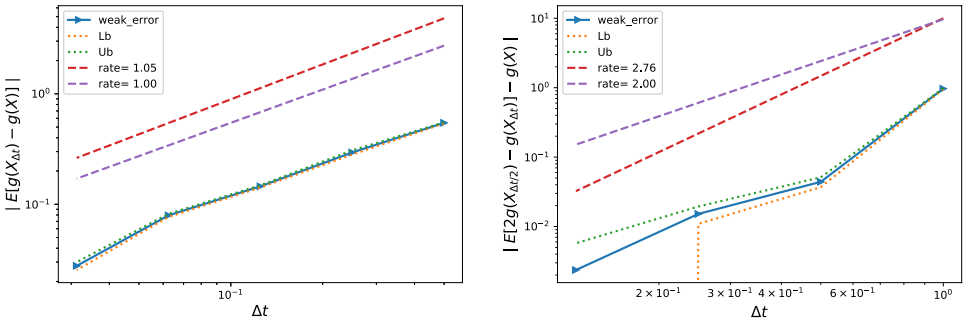
Parameters	Reference solution
Set 1: $H = 0.07, K = 1, S_0 = 1, T = 1, \rho = -0.9, \eta = 1.9, \xi_0 = 0.235^2$	0.0791 $(7.9e{-}05)$
Set 2: $H = 0.02, K = 1, S_0 = 1, T = 1, \rho = -0.7, \eta = 0.4, \xi_0 = 0.1$	0.1248 $(1.3e{-}04)$
Set 3: $H = 0.02, K = 0.8, S_0 = 1, T = 1, \rho = -0.7, \eta = 0.4, \xi_0 = 0.1$	0.2407 $(5.6e{-}04)$
Set 4: $H = 0.02, K = 1.2, S_0 = 1, T = 1, \rho = -0.7, \eta = 0.4, \xi_0 = 0.1$	0.0568 $(2.5e{-}04)$

Table 5.1: Reference solution, which is the approximation of the call option price under the rBergomi model, defined in (2.6), using MC with 500 time steps and number of samples, $M = 10^6$, for different parameter constellations. The numbers between parentheses correspond to the statistical errors estimates.

5.1 Weak error

We start our numerical experiments with accurately estimating the weak error (bias) for the different parameter sets in Table 5.1, with and without Richardson extrapolation.

For illustrative purposes, we only show the weak errors related to set 1 in Table 5.1 (see Figure 5.1). We note that we observed similar behavior for the other parameter sets, with slightly worse rates for some cases. We emphasize that the reported weak rates correspond to the pre-asymptotic regime that we are interested in. We are not interested in estimating the rates specifically but rather obtaining a sufficiently precise estimate of the weak error (bias), $\mathcal{E}_B(N)$, for different numbers of time steps N. For a fixed discretization, the corresponding estimated biased solution will be set as a reference solution to the ASGQ method in order to estimate the quadrature error $\mathcal{E}_Q(\mathrm{TOL}_{\mathrm{ASGQ}}, N)$.



(a) (b)

Figure 5.1: The convergence of the weak error $\mathcal{E}_B(N)$, defined in (4.2), using MC, for set 1 parameter in Table 5.1. We refer to C_{RB} as $\text{E}[g(X)]$, and to C^N_{RB} as $\text{E}[g(X_{\Delta t})]$. The upper and lower bounds are 95% confidence intervals. a) without Richardson extrapolation. b) with Richardson extrapolation (level 1).

5.2 Comparing the different errors and computational time for MC and ASGQ

In this section, we conduct a comparison between MC and ASGQ in terms of errors and computational time. We show tables and plots reporting the different relative errors involved in the MC method (bias and statistical error[4] estimates), and in ASGQ (bias and quadrature error estimates). While fixing a sufficiently small error tolerance in the price estimates, we also compare the computational time needed for both methods to meet the desired error tolerance. We note that in all cases the actual work (runtime) is obtained using an Intel(R) Xeon(R) CPU E5-268 architecture.

Through our conducted numerical experiments for each parameter set, we follow these steps to achieve our reported results:

i) For a fixed number of time steps, N, we compute an accurate estimate, using a large number of samples, M, of the biased MC solution, C^N_{RB}. This step also provides us with an estimate of the bias error, $\mathcal{E}_B(N)$, defined by (4.2).

ii) The estimated biased solution, C^N_{RB}, is used as a reference solution to the ASGQ method to compute the quadrature error, $\mathcal{E}_Q(\text{TOL}_{\text{ASGQ}}, N)$, defined by (4.4).

iii) In order to compare with MC method, the number of samples, M, is chosen so that the statistical error of the Monte Carlo method, $\mathcal{E}_S(M)$, satisfies

$$\mathcal{E}_S(M) = \mathcal{E}_B(N) = \frac{\mathcal{E}_{\text{tot}}}{2}, \tag{5.1}$$

where $\mathcal{E}_B(N)$ is the bias as defined in (4.2) and $\mathcal{E}_{\text{tot}}$ is the total error.

We show the summary of our numerical findings in Table 5.2, which highlights the computational gains achieved by ASGQ over MC method to meet a certain error tolerance, which we set

[4]The statistical error estimate of MC is $C_\alpha \frac{\sigma_M}{\sqrt{M}}$, where M is the number of samples and $C_\alpha = 1.96$ for 95% confidence interval.

approximately to 1%. More detailed results for each case of parameter set, as in Table 5.1, are provided in Sections 5.2.1, 5.2.2, 5.2.3 and 5.2.4.

Parameter set	Level of Richardson extrapolation	Total relative error	Ratio of CPU time (MC/ASGQ)
Set 1	level 1	3%	1.6
	level 2	1%	> 9
Set 2	without	0.2%	5
Set 3	without	0.4%	7
Set 4	without	2%	1.3

Table 5.2: Summary of relative errors and computational gains, achieved by the different methods. In this table, we highlight the computational gains achieved by ASGQ over MC method to meet a certain error tolerance. As expected, these gains are improved when applying Richardson extrapolation as observed for the case of parameters set 1. We provide details about the way we compute these gains for each case in the following sections.

5.2.1 Case of parameters in Set 1, in Table 5.1

In this section, we conduct our numerical experiments for three different scenarios: i) without Richardson extrapolation (see Tables 5.3 and 5.4), ii) with (level 1) Richardson extrapolation (see Tables 5.5 and 5.6), and iii) with (level 2) Richardson extrapolation (see Tables 5.7 and 5.8). Our numerical experiments show that ASGQ coupled with (level 1) Richardson extrapolation requires approximately 60% of the work of MC coupled with (level 1) Richardson extrapolation, to achieve a total relative error of around 3%. This gain is improved further when applying level 2 Richardson extrapolation. In fact, ASGQ coupled with (level 2) Richardson extrapolation requires approximately less than 10% of the work of MC coupled with (level 2) Richardson extrapolation, to achieve a total relative error below 1%. Applying Richardson extrapolation brought a significant improvement for ASGQ (see Figure 5.2 and Tables 5.3,5.4,5.5,5.6,5.7,5.8).

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	**0.69** (0.54,0.15)	**0.42** (0.29,0.13)	**0.31** (0.15,0.16)	**0.11** (0.07,0.04)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	**0.66** (0.54,0.12)	**0.29** (0.29,6e−04)	**0.16** (0.15,0.01)	**0.08** (0.07,0.01)
QMC	**1.13** (0.54,0.59)	**0.635** (0.295,0.34)	**0.345** (0.155,0.19)	**0.17** (0.07,0.10)
M(# QMC samples)	$2^3 \times 2^3 = 64$	$2^3 \times 2^4 = 128$	$2^3 \times 2^5 = 256$	$2^3 \times 2^6 = 512$
MC	**1.05** (0.54,0.51)	**0.59** (0.295,0.295)	**0.31** (0.155,0.155)	**0.14** (0.07,0.07)
M(# MC samples)	2×10	4×10	10^2	4×10^2

Table 5.3: Total relative error of ASGQ, without Richardson extrapolation, with different tolerances, and MC to compute the call option prices for different numbers of time steps. The values between parentheses correspond to the different errors contributing to the total relative error: for ASGQ we report the bias and quadrature errors and for MC we report the bias and the statistical errors estimates. The number of MC samples, M, is chosen to satisfy (5.1).

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	0.08	0.13	0.7	163
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	0.2	5	333	1602
QMC method	0.006	0.012	0.026	0.064
MC method	0.001	0.003	0.02	0.2

Table 5.4: Comparison of the computational time (in seconds) of MC and ASGQ, to compute the call option price of the rBergomi model for different numbers of time steps. The average MC CPU time is computed over 100 runs.

Method	Steps		
	$1-2$	$2-4$	$4-8$
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	**1.33** (0.96,0.37)	**0.18** (0.07,0.11)	**0.144** (0.015,0.129)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 5.10^{-2}$)	**1.33** (0.96,0.37)	**0.23** (0.07,0.16)	**0.025** (0.015,0.010)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	**1.08** (0.96,0.12)	**0.08** (0.07,0.01)	**0.025** (0.015,0.010)
QMC	**1.59** (0.96,0.63)	**0.14** (0.07,0.07)	**0.031** (0.015,0.016)
M(# QMC samples)	$2^3 \times 2^4 = 128$	$2^3 \times 2^{10} = 8192$	$2^3 \times 2^{13} = 65536$
MC	**1.88** (0.96,0.92)	**0.14** (0.07,0.07)	**0.03** (0.015,0.015)
M(# MC samples)	10	2×10^3	4×10^4

Table 5.5: Total relative error of ASGQ, coupled with Richardson extrapolation (level 1), with different tolerances, and MC, coupled with Richardson extrapolation (level 1), to compute the call option price for different numbers of time steps. The values between parentheses correspond to the different errors contributing to the total relative error: for ASGQ we report the bias and quadrature errors and for MC we report the bias and the statistical errors. The number of MC samples, M, is chosen to satisfy (5.1). The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

Method	Steps		
	$1-2$	$2-4$	$4-8$
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	0.1	0.2	1.6
ASGQ ($\text{TOL}_{\text{ASGQ}} = 5.10^{-2}$)	0.1	0.6	37
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	1.3	6	2382
QMC	0.02	1	10
MC	0.003	2	60

Table 5.6: Comparison of the computational time (in seconds) of MC and ASGQ, using Richardson extrapolation (level 1), to compute the call option price of the rBergomi model for different numbers of time steps. The average MC CPU time is computed over 100 runs. The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

Method	Steps	
	$1-2-4$	$2-4-8$
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	**0.54** (0.24,0.30)	**0.113** (0.006,0.107)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 5.10^{-2}$)	**0.49** (0.24,0.25)	**0.009** (0.006,0.003)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	**0.27** (0.24,0.03)	**0.009** (0.006,0.003)
QMC	**0.5** (0.24,0.26)	**0.014** (0.006,0.008)
M(# QMC samples)	$2^3 \times 2^7 = 1024$	$2^3 \times 2^{16} = 524288$
MC	**0.45** (0.24,0.21)	**0.012** (0.006,0.006)
M(# MC samples)	4×10^2	4×10^5

Table 5.7: Total relative error of ASGQ, coupled with Richardson extrapolation (level 2), with different tolerances, and MC, coupled with Richardson extrapolation (level 2), to compute the call option price for different numbers of time steps. The values between parentheses correspond to the different errors contributing to the total relative error: for ASGQ we report the bias and quadrature errors and for MC we report the bias and the statistical errors. The number of MC samples, M, is chosen to satisfy (5.1). The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

Method	Steps	
	$1-2-4$	$2-4-8$
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	0.2	2
ASGQ ($\text{TOL}_{\text{ASGQ}} = 5.10^{-2}$)	0.5	74
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	9	3455
QMC	0.2	110
MC	0.2	690

Table 5.8: Comparison of the computational time (in seconds) of MC and ASGQ, using Richardson extrapolation (level 2), to compute the call option price of the rBergomi model for different numbers of time steps. The average MC CPU time is computed over 100 runs. The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

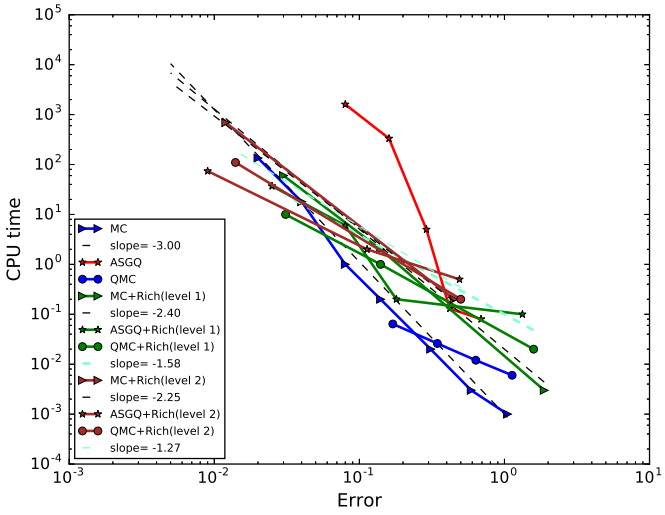


Figure 5.2: Computational work comparison for ASGQ and MC methods (with and without) Richardson extrapolation, for the case of parameter set 1 in Table 5.1. This plot shows that to achieve a relative error below 1%, ASGQ coupled with level 2 of Richardson extrapolation is the best option in terms of computational time. Furthermore, applying Richardson extrapolation brings a significant improvement for ASGQ and MC methods, in terms of numerical complexity.

5.2.2 Case of parameters in Set 2, in Table 5.1

In this section, we only conduct our numerical experiments for the case without Richardson extrapolation, since the results show that we meet a small enough error tolerance without the need to apply Richardson extrapolation. Our numerical experiments show that ASGQ requires approximately 20% of the work of MC method, to achieve a total relative error of around 0.2% (see Figure 5.3 and Tables 5.10 and 5.9).

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	**0.03** (0.02,0.01)	**0.022** (0.008,0.014)	**0.022** (0.004,0.018)	**0.017** (0.001,0.016)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	**0.03** (0.02,0.01)	**0.017** (0.008,0.009)	**0.008** (0.004,0.004)	**0.001** (0.001,4e−04)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-3}$)	**0.02** (0.02,8e−04)	**0.009** (0.008,8e−04)	**0.005** (0.004,8e−04)	**0.001** (0.001,4e−04)
QMC	**0.04** (0.02,0.02)	**0.016** (0.008,0.008)	**0.009** (0.004,0.005)	**0.0025** (0.001,0.0015)
M(# QMC samples)	$2^3 \times 2^8 = 2048$	$2^3 \times 2^{10} = 8192$	$2^3 \times 2^{11} = 16384$	$2^3 \times 2^{13} = 65536$
MC	**0.04** (0.02,0.02)	**0.016** (0.008,0.008)	**0.007** (0.004,0.003)	**0.002** (0.001,0.001)
M(# MC samples)	4×10^3	2×10^4	10^5	10^6

Table 5.9: Total relative error of ASGQ, without Richardson extrapolation, with different tolerances, and MC to compute the call option price for different numbers of time steps. The values between parentheses correspond to the different errors contributing to the total relative error: for ASGQ we report the bias and quadrature errors and for MC we report the bias and the statistical errors estimates. The number of MC samples, M, is chosen to satisfy (5.1). The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	0.1	0.1	0.2	0.8
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	0.1	0.5	8	92
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-3}$)	0.5	3	24	226
QMC method	0.2	0.7	1.6	8
MC method	0.15	1.6	16.5	494

Table 5.10: Comparison of the computational time (in seconds) of MC and ASGQ, to compute the call option price of the rBergomi model for different numbers of time steps. The average MC CPU time is computed over 100 runs. The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

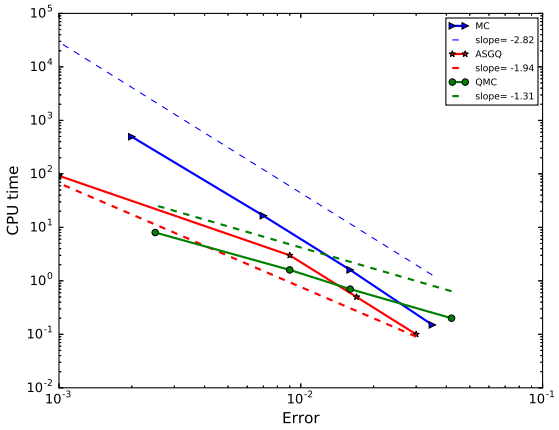


Figure 5.3: Computational work comparison for ASGQ and MC methods, for the case of parameter set 2 in Table 5.1. This plot shows that to achieve a relative error below 1%, ASGQ outperforms MC method in terms of computational time.

5.2.3 Case of parameters in Set 3, in Table 5.1

In this section, we only conduct our numerical experiments for the case without Richardson extrapolation, since the results show that we meet a small enough error tolerance without the need to apply Richardson extrapolation. Our numerical experiments show that ASGQ requires approximately 14% of the work of MC method, to achieve a total relative error of around 0.4% (see Figure 5.4 and Tables 5.12 and 5.11).

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	**0.008** (0.006,0.002)	**0.009** (0.004,0.005)	**0.008** (0.003,0.005)	**0.009** (0.002,0.007)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	**0.008** (0.006,0.002)	**0.009** (0.004,0.005)	**0.005** (0.003,0.002)	**0.002** (0.002,$1e{-}04$)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-3}$)	**0.008** (0.006,0.002)	**0.006** (0.004,0.002)	**0.003** (0.003,$1e{-}04$)	**0.002** (0.002,$1e{-}04$)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-4}$)	**0.006** (0.006,$4e{-}04$)	**0.004** (0.004,$2e{-}04$)	**0.003** (0.003,$1e{-}04$)	–
QMC	**0.012** (0.006,0.006)	**0.0086** (0.004,0.0046)	**0.0062** (0.003,0.0032)	**0.0038** (0.002,0.0018)
M(# QMC samples)	$2^3 \times 2^{10} = 8192$	$2^3 \times 2^{10} = 8192$	$2^3 \times 2^{11} = 16384$	$2^3 \times 2^{12} = 32768$
MC	**0.01** (0.006,0.005)	**0.008** (0.004,0.004)	**0.006** (0.003,0.003)	**0.004** (0.002,0.002)
M(# MC samples)	2×10^4	4×10^4	6×10^4	8×10^4

Table 5.11: Total relative error of ASGQ, without Richardson extrapolation, with different tolerances, and MC to compute the call option price for different numbers of time steps. The values between parentheses correspond to the different errors contributing to the total relative error: for ASGQ we report the bias and quadrature errors and for MC we report the bias and the statistical errors estimates. The number of MC samples, M, is chosen to satisfy (5.1). The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	0.1	0.1	0.1	1
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	0.1	0.15	9	112
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-3}$)	0.2	2	27	2226
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-4}$)	1	6	136	–
QMC method	0.6	0.7	1.6	4
MC method	1	3	10	40

Table 5.12: Comparison of the computational time (in seconds) of MC and ASGQ, to compute the call option price of the rBergomi model for different numbers of time steps. The average MC CPU time is computed over 100 runs. The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

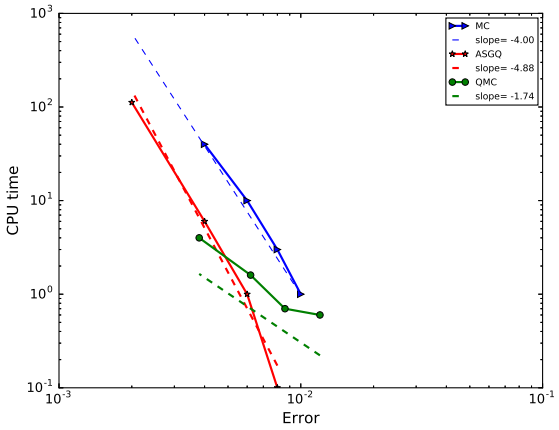


Figure 5.4: Comparison of computational work for MC and ASGQ methods, for the case of parameter set 3 in Table 5.1. This plot shows that to achieve a relative error below 1%, ASGQ outperforms MC method in terms of computational time.

5.2.4 Case of parameters in Set 4, in Table 5.1

In this section, we only conduct our numerical experiments for the case without Richardson extrapolation. Our numerical experiments show that ASGQ requires approximately 75% of the work of MC method, to achieve a total relative error of around 2% (see Figure 5.5 and Tables 5.14 and 5.13). Similar to the case of set 1 parameters illustrated in section 5.2.1, we believe that Richardson extrapolation will improve the performance of ASGQ method.

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	**0.09** (0.07,0.05)	**0.07** (0.03,0.04)	**0.07** (0.02,0.05)	**0.06** (0.01,2e−04)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	**0.09** (0.07,5e−04)	**0.07** (0.03,0.04)	**0.02** (0.02,3e−04)	**0.02** (0.01,2e−04)
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-3}$)	**0.07** (0.07,5e−04)	**0.03** (0.03,4e−04)	**0.02** (0.02,3e−04)	**0.01** (0.01,2e−04)
QMC	**0.135** (0.07,0.065)	**0.67** (0.03,0.03)	**0.044** (0.02,0.024)	**0.022** (0.01,0.012)
M(# QMC samples)	$2^3 \times 2^7 = 1024$	$2^3 \times 2^8 = 2048$	$2^3 \times 2^9 = 4096$	$2^3 \times 2^10 = 8192$
MC	**0.14** (0.07,0.07)	**0.07** (0.03,0.04)	**0.04** (0.02,0.02)	**0.02** (0.01,0.01)
M(# MC samples)	6×10^2	2×10^3	8×10^3	2×10^4

Table 5.13: Total relative error of ASGQ, without Richardson extrapolation, with different tolerances, and MC to compute the call option price for different numbers of time steps. The values between parentheses correspond to the different errors contributing to the total relative error: for ASGQ we report the bias and quadrature errors and for MC we report the bias and the statistical errors estimates. The number of MC samples, M, is chosen to satisfy (5.1). The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

Method	Steps			
	2	4	8	16
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-1}$)	0.1	0.1	0.2	0.5
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-2}$)	0.1	0.1	8	97
ASGQ ($\text{TOL}_{\text{ASGQ}} = 10^{-3}$)	0.7	4	26	1984
QMC method	0.08	0.17	0.4	1
MC method	0.02	0.15	1.4	10

Table 5.14: Comparison of the computational time (in seconds) of MC and ASGQ, to compute the call option price of rBergomi model for different numbers of time steps. The average MC CPU time is computed over 100 runs. The values marked in red correspond to the values used for computational work comparison against MC method, reported in Table 5.2.

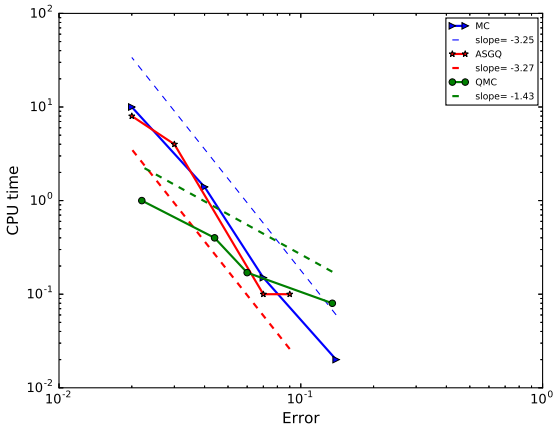


Figure 5.5: Comparison of computational work for MC and ASGQ methods, for the case of parameter set 4 in Table 5.1. This plot shows that to achieve a relative error around 1%, ASGQ and MC methods have similar performance in terms of computational time.

6 Conclusions and future work

In this work, we propose a novel, fast option pricer, based on a hierarchical ASGQ, specifically using the same construction as in [22], coupled with Brownian bridge construction and Richardson extrapolation, for options whose underlyings follow the rBergomi model as in [4].

Given that the only prevalent option, in this context, is to use different variants of the MC method, which is computationally expensive, our first contribution is that we uncover the available regularity in the rBergomi model and design a novel alternative approach based on an ASGQ. This approach opens a new research direction in this field to investigate the performance of other methods besides MC, for pricing and calibrating under the rBergomi model. Our second contribution is that we reduce the computational cost through bias reduction by using Richardson extrapolation. Finally, assuming one targets price estimates with a sufficiently small error tolerance, our proposed method demonstrates substantial computational gains over the standard MC method, when pricing under the rBergomi model, even for very small values of the Hurst parameter. We show these gains through our numerical experiments for different parameter constellations. We clarify that we do not claim that these gains will hold in the asymptotic regime, i.e. for higher accuracy requirements. Furthermore, the use of Richardson extrapolation is justified in the pre-asymptotic regime, in which our observed experimental results suggest a convergence of order one for the weak error. We emphasize that, to the best of our knowledge, no proper weak error analysis has been done in the rough volatility context.

In this work, we limit ourselves to compare our novel proposed method against the standard MC. A more systematic comparison against the variant of MC proposed in [28] can be carried out but this remains for a future study. Another potential direction of future research may also investigate the performance of QMC for such problems. Finally, accelerating our novel method can be achieved by coupling ASGQ with a more optimal hierarchical path generation method than Brownian bridge construction, such as PCA or LT transformations.

Acknowledgments C. Bayer gratefully acknowledges support from the German Research Foundation (DFG, grant BA5484/1). This work was supported by the KAUST Office of Sponsored Research (OSR) under Award No. URF/1/2584-01-01 and the Alexander von Humboldt Foundation. C. Ben Hammouda and R. Tempone are members of the KAUST SRI Center for Uncertainty Quantification in Computational Science and Engineering. The authors would like to thank Joakim Beck, Eric Joseph Hall and Erik von Schwerin for their helpful and constructive comments that greatly contributed to improving the final version of the paper.

References Cited

[1] Peter A Acworth, Mark Broadie, and Paul Glasserman. A comparison of some Monte Carlo and quasi Monte Carlo techniques for option pricing. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, pages 1–18. Springer, 1998.

[2] Elisa Alòs, Jorge A León, and Josep Vives. On the short-time behavior of the implied volatility for jump-diffusion models with stochastic volatility. *Finance and Stochastics*, 11(4):571–589, 2007.

[3] Pierre Bajgrowicz, Olivier Scaillet, and Adrien Treccani. Jumps in high-frequency data: Spurious detections, dynamics, and news. *Management Science*, 62(8):2198–2217, 2015.

[4] Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.

[5] Christian Bayer, Peter K Friz, Paul Gassiat, Joerg Martin, and Benjamin Stemper. A regularity structure for rough volatility. *arXiv preprint arXiv:1710.07481*, 2017.

[6] Christian Bayer, Peter K Friz, Archil Gulisashvili, Blanka Horvath, and Benjamin Stemper. Short-time near-the-money skew in rough fractional volatility models. *Quantitative Finance*, pages 1–20, 2018.

[7] CHRISTIAN BAYER, MARKUS SIEBENMORGEN, and RAUL TEMPONE. Smoothing the payoff for efficient computation of basket option pricing.

[8] Mikkel Bennedsen, Asger Lunde, and Mikko S Pakkanen. Decoupling the short-and long-term behavior of stochastic volatility. *arXiv preprint arXiv:1610.00332*, 2016.

[9] Mikkel Bennedsen, Asger Lunde, and Mikko S Pakkanen. Hybrid scheme for Brownian semistationary processes. *Finance and Stochastics*, 21(4):931–965, 2017.

[10] Lorenzo Bergomi. Smile dynamics II. *Risk*, 18:67–73, 2005.

[11] F. Biagini, Y. Hu, B. Øksendal, and T. Zhang. *Stochastic Calculus for Fractional Brownian Motion and Applications*. Probability and Its Applications. Springer London, 2008.

[12] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[13] Russel E Caflisch, William J Morokoff, and Art B Owen. *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension*. 1997.

[14] Kim Christensen, Roel CA Oomen, and Mark Podolskij. Fact or friction: Jumps at ultra high frequency. *Journal of Financial Economics*, 114(3):576–599, 2014.

[15] Ronald Cools and Dirk Nuyens. A belgian view on lattice rules. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 3–21. Springer, 2008.

[16] Laure Coutin. An introduction to (stochastic) calculus with respect to fractional Brownian motion. In *Séminaire de Probabilités XL*, pages 3–65. Springer, 2007.

[17] Martin Forde and Hongzhong Zhang. Asymptotics for rough stochastic volatility models. *SIAM Journal on Financial Mathematics*, 8(1):114–145, 2017.

[18] Masaaki Fukasawa. Asymptotic analysis for stochastic volatility: martingale expansion. *Finance and Stochastics*, 15(4):635–654, 2011.

[19] Jim Gatheral, Thibault Jaisson, Andrew Lesniewski, and Mathieu Rosenbaum. Volatility is rough, part 2: Pricing. Workshop on Stochastic and Quantitative Finance, Imperial College London, London, 2014.

[20] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative Finance*, 18(6):933–949, 2018.

[21] Paul Glasserman. *Monte Carlo methods in financial engineering*. Springer, New York, 2004.

[22] Abdul-Lateef Haji-Ali, Fabio Nobile, Lorenzo Tamellini, and Raul Tempone. Multi-index stochastic collocation for random pdes. *Computer Methods in Applied Mechanics and Engineering*, 306:95–122, 2016.

[23] Junichi Imai and Ken Seng Tan. Minimizing effective dimension using linear transformation. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 275–292. Springer, 2004.

[24] Antoine Jacquier, Claude Martini, and Aitor Muguruza. On VIX futures in the rough Bergomi model. *Quantitative Finance*, 18(1):45–61, 2018.

[25] Antoine Jacquier, Mikko S Pakkanen, and Henry Stone. Pathwise large deviations for the rough Bergomi model. *arXiv preprint arXiv:1706.05291*, 2017.

[26] Benoit B Mandelbrot and John W Van Ness. Fractional Brownian motions, fractional noises and applications. *SIAM review*, 10(4):422–437, 1968.

[27] Domenico Marinucci and Peter M Robinson. Alternative forms of fractional Brownian motion. *Journal of statistical planning and inference*, 80(1-2):111–122, 1999.

[28] Ryan McCrickerd and Mikko S Pakkanen. Turbocharging Monte Carlo pricing for the rough Bergomi model. *Quantitative Finance*, pages 1–10, 2018.

[29] William J Morokoff and Russel E Caflisch. Quasi-random sequences and their discrepancies. *SIAM Journal on Scientific Computing*, 15(6):1251–1279, 1994.

[30] Bradley Moskowitz and Russel E Caflisch. Smoothness and dimension reduction in quasi-Monte Carlo methods. *Mathematical and Computer Modelling*, 23(8):37–54, 1996.

[31] Andreas Neuenkirch and Taras Shalaiko. The order barrier for strong approximation of rough volatility models. *arXiv preprint arXiv:1606.03854*, 2016.

[32] Dirk Nuyens. The construction of good lattice rules and polynomial lattice rules., 2014.

[33] Jean Picard. Representation formulae for the fractional Brownian motion. In *Séminaire de Probabilités XLIII*, pages 3–70. Springer, 2011.

[34] Marc Romano and Nizar Touzi. Contingent claims and market completeness in a stochastic volatility model. *Mathematical Finance*, 7(4):399–412, 1997.

[35] Ian H Sloan. Lattice methods for multiple integration. *Journal of Computational and Applied Mathematics*, 12:131–143, 1985.

[36] Denis Talay and Luciano Tubaro. Expansion of the global error for numerical schemes solving stochastic differential equations. *Stochastic analysis and applications*, 8(4):483–509, 1990.