Weak error of rough volatility models

Jens Nierste
June 2018

Technical University Berlin
Institute for Mathematics

Supervised by
Prof. Dr. Peter K. Friz
TU Berlin

Assistant Supervisor
Dr. Christian Bayer
WIAS Institute Berlin

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

Berlin,

Abstract

This thesis is about the weak error of the so called rough volatility models. Those models are an extension of the classical stochastic volatility models. Instead of using a normal Brownian motion W to simulate the volatility process those models use a fractional Brownian motion W^H with Hurst parameter $H < \frac{1}{2}$. Those kind of models were introduced by Gatheral et al [15]. C. Bayer et al [4] introduced the rough Bergomi model, which gets examined in this thesis.
For the rough Bergomi model exist two different approximation schemes:

- an exact but time-consuming scheme
- the hybrid scheme [8], which needs less time for the computation.

Our goal is to determine for both schemes with the help of Monte Carlo simulations an order of convergence for the weak error. We expect thereby that the exact scheme has a higher order of convergence than the hybrid scheme.
Before we start with the introduction of the rough volatility models we repeat the definition and properties of the Itô integral for Brownian motions and the Riemann-Stieltjes integral.
Then we introduce the rough volatility models and repeat the definition of the fractional Brownian motion and some of it's properties. Because we can't use the Itô integral for the integration with respect to the fractional Brownian motion we have to introduce the Young integral, which allows us to integrate with respect to the fractional Brownian motion.
Then we present some general theory of the approximation of stochastic differential equations and Monte Carlo simulations.
Before we start with the analysis of the weak error of the rough Bergomi model, we present a result for the strong error of rough volatility models.
Thereafter we introduce the two approximation schemes for the rough Bergomi model: the exact and the hybrid scheme.
Before we start with our numerical experiments we talk about our implementation of the schemes, i.e. what kind of functions we used in our code.
Finally we use numerical experiments and simulations to determine an order of convergence for the exact and hybrid scheme. Thereby we get the results that the exact scheme has an order of convergence of 1 and the hybrid scheme has an order of convergence of 0.5.
Furthermore we analyse the effect of the parameter κ of the hybrid scheme. In [8] it was already detected that $\kappa = 0$ doesn't convergence to the correct solution and that $\kappa = 1, \ldots, 3$ are not distinguishable. We examine the effect in terms of the weak error. Thereby we also conclude that for $\kappa = 0$

the hybrid scheme doesn't convergence to the correct result. Furthermore we determine that for $\kappa = 1, \ldots, 3$ we get a difference of the order 10^{-4}. Against our expectations we got the best result from $\kappa = 1$.
Finally the appendix contains our Python code for the exact and the hybrid scheme.

Zusammenfassung

Diese Masterarbeit beschäftigt sich mit dem schwachen Fehler von sogenannten rough volatility models, was sich am besten mit rauen Volatilitätsmodellen übersetzen lässt, genauer gesagt mit dem schwachen Fehler des rough Bergomi Modells.
Diese Modelle sind eine Erweiterung von klassischen stochastischen Volatilitätsmodellen. Anstatt von einer klassischen Brownschen Bewegung W, wird die Volatilität mithilfe einer gebrochenen Brownschen Bewegung W^H mit Hurst Parameter $H < \frac{1}{2}$ modelliert. Diese Art von Modellen wurde von Gatheral et al [15] eingeführt. C. Bayer et al [4] haben das rough Bergomi Model entwickelt, welches in dieser Arbeit untersucht wird.
Für das rough Bergomi Model gibt es zwei Approximationstechniken:

- ein exaktes, aber zeitaufwendiges Approximationsschema
- das Hybrid-Schema [8], welches weniger Zeit für die Berechnung benötigt.

Unser Ziel ist es, für beide Schemata mittels Monte Carlo Simulation für den schwachen Fehler eine Konvergenzordnung zu bestimmen. Wir erwarten dabei, dass das exakte Schema eine höhere Konvergenzordnung als das Hybrid-Schema hat.
Bevor wir mit der Einführung der rauen Volatilitätsmodelle anfangen, wiederholen wir die Definition und ein paar Eigenschaften des Itô Integrals bzgl. der Brownschen Bewegung und der Riemann-Stieltjes Integrale.
Danach stellen wir die rauen Volatilitätsmodelle vor und wiederholen die Definition der gebrochenen Brownschen Bewegung und einige ihrer Eigenschaften. Im Folgenden führen wir das Young Integral ein, welches uns ermöglicht bzgl. der gebrochenen Brownschen Bewegung zu integrieren, da wir das Itô Integral dafür nicht nutzen können.
Danach präsentieren wir die allgemeine Theorie zur numerischen Approximation für stochastische Differentialgleichungen und zu Monte Carlo Simulationen.
Bevor wir mit der Analyse des schwachen Fehlers des rauen Bergomi Modells anfangen, präsentieren wir noch ein Resultat bezüglich des starken Fehlers von rauen Volatilitätsmodellen.
Anschließend stellen wir die zwei Approximations-Schemata für das raue Bergomi Modell vor: das exakte und das Hybrid-Schema.
Bevor wir mit unseren numerischen Experimenten anfangen, gehen wir noch kurz auf die Implementierung der Schemata ein, d.h. auf die besonderen Funktionen bzw. Methoden, die wir in der Implementierung nutzen.

Schlussendlich benutzen wir numerische Experimente und Simulationen, um eine Konvergenzordnung für das exakte und das Hybrid-Schema zu bestimmen. Dabei stellen wir für das exakte Schema eine Konvergenzordnung von 1 fest und für das Hybrid-Schema von 0,5.
Außerdem untersuchen wir den Effekt des Parameters κ im Hybrid-Schema. Es wurde in [8] bereits festgestellt, dass das Hybrid-Schema für die implizite Volatilität für $\kappa = 0$ nicht die gewünschten Ergebnisse liefert und für $\kappa = 1, \ldots, 3$ nicht vom exakten Schema zu unterscheiden ist. Wir untersuchen diesen Effekt nochmals in Bezug auf den schwachen Fehler. Dabei stellen wir ebenfalls fest, dass das Hybrid-Schema für $\kappa = 0$ nicht gegen die korrekte Lösung konvergiert. Des Weiteren konnten wir feststellen, dass sich die Ergebnisse von $\kappa = 1, \ldots, 3$ marginal im Bereich 10^{-4} unterscheiden. Entgegen unseren Erwartungen lieferte $\kappa = 1$ sogar das beste Ergebnis.
Zuallerletzt enthält der Anhang noch unseren Python Code des exakten und des Hybrid-Schemas.

Contents

List of Figures

List of Tables

1 Introduction

In financial mathematics the pricing of derivatives is an important topic. Today all kinds of derivatives are traded like puts and calls or more complex derivatives like Asian Options. The pricing is done by the calculation of financial models with Monte Carlo simulations.
In finance mathematics stock prices S_t are modelled in form of stochastic differential equations like

$$dS_t = \mu_t dt + \sigma_t S_t dW_t,$$

where μ_t is a drift term, W_t is a one-dimensional Brownian motion and σ_t is the volatility process.
If the volatility process $\sigma_t = \sigma$ is a constant, then we receive the classical Black-Scholes model

$$dS_t = \mu_t dt + \sigma S_t dW_t.$$

The problem of the Black-Scholes model is that in reality the assumption that σ is constant doesn't hold. Let's therefore take a look at the Black-Scholes formula for a call-price $e^{-r}(S_T - K)^+$, where S_T is given by the Black-Scholes model, K is the strike price and r is the risk free interest rate. Then the Black-Scholes formula of the call-price is given by

$$BS(S_0, K, r, T, \sigma) = S_0\Phi(d_+) - Ke^{-rT}\Phi(d_-),$$

with

$$d_\pm = \frac{\log(\frac{S_0}{K}) + (r \pm \frac{1}{2}\sigma^2)}{\sigma\sqrt{T}}$$

and $\Phi(x)$ denotes the cumulative distribution function of the standard normal distribution [22].
Every parameter of the Black-Scholes formula besides σ can be observed at the market. Thus if we take market data for a option price $\tilde{C}$ and the underlying parameters $(\tilde{S}_0, \tilde{K}, \tilde{r}, \tilde{T})$ we can calculate the volatility which is needed so that the price of the Black-Scholes formula fits the observed market price of the option. This is called the implied volatility $\sigma_{imp}(\tilde{S}_0, \tilde{K}, \tilde{r}, \tilde{T})$

$$C = BS(\tilde{S}_0, \tilde{K}, \tilde{r}, \tilde{T}, \sigma_{imp}(\tilde{S}_0, \tilde{K}, \tilde{r}, \tilde{T})).$$

If we use the implied volatility to calculate the volatility from the market prices, we see that the volatility depends on the maturity T and the strike price K of the option. Hence one obtains the famous volatility smile [4, Figure 1.1]

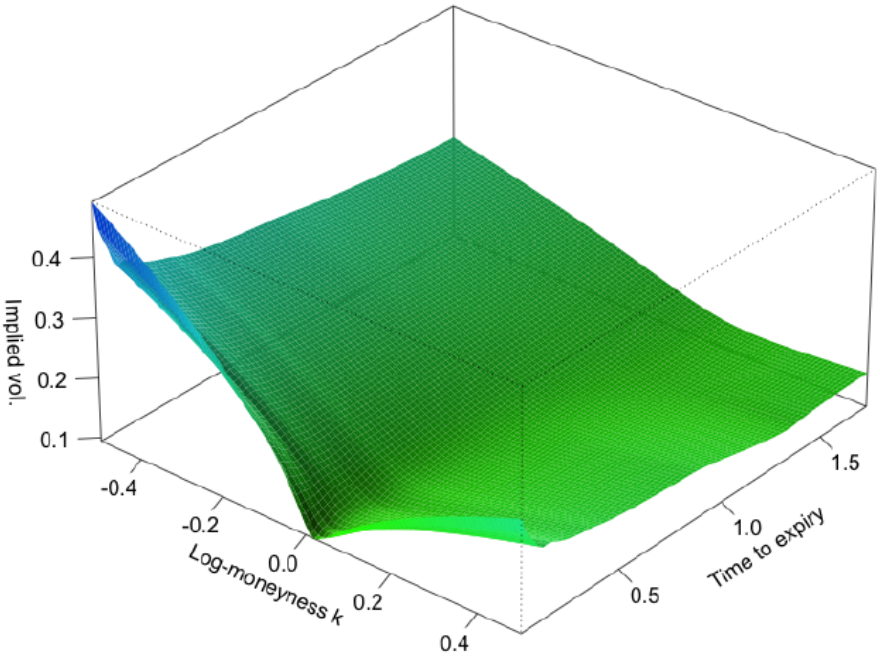

Figure 1: The SPX volatility surface as of August 14, 2013. Time measured in years.

Therefore further models developed, where the volatility isn't a constant but either a deterministic function $\sigma(S_t, t)$ or a stochastic process, where σ_t is again a random process given by stochastic differential equation with respect to a Brownian motion.
The models where the volatility is given by a deterministic function are called local volatility models.
If the volatility is given by a stochastic process the model is called a stochastic volatility model. Classical stochastic volatility models are for example the Heston, SABER and Stein-Stein model [6].

Recently a new type of stochastic volatility models was introduced by J. Gatheral et al in 2014 [15]: Rough volatility models. Instead of a Brownian motion they use a fractional Brownian motion [28, Def. 2.1] with Hurst parameter $H < \frac{1}{2}$ for the modelling of the volatility process. As shown in [15] and [4] those models are remarkably consistent with time series data.
In this thesis we want to take a closer look at the weak approximation error,i.e.

$$|\mathbb{E}[f(X_T)] - \mathbb{E}[f(X_T^N)]|,$$

of rough volatility models, especially the rough Bergomi model (for the exact definition of the weak error see 5.2.2).
So far there are no results on the speed of convergence in the weak error sense for the rBergomi model. Our goal is to use numerical simulations to determine the weak approximation error of the rBergomi model and get first results on a possible convergence rate.
Because no analytical solution for the rBergomi model exists we can't compare our simulation results with an exact solution to determine the weak approximation error. Instead we need to use Monte Carlo simulations with a high number of repetitions and a small step-size to get a good approximation solution. Thus the handling of the computational effort is important in the simulations, which can get quite challenging.
If we want to use an exact simulation like Bayer et al [4] we have to calculate huge covariance matrices for the generation of the random variables. To calculate those alone takes a lot of time.
Recently a new simulation method for semi-stationary Brownian motion called the hybrid scheme [8] was developed. As we will see the volatility process in the rBergomi model is a semi-stationary Brownian motion thus it can be modelled using the hybrid scheme. Compared to the exact simulation approach of Bayer et al [4] it is not necessary to calculate huge covariance matrices. Thus the computational effort is lower and it became the go to method for simulating the rBergomi model. Nevertheless the computation still takes quite some time.
For receiving simulation results we use Monte Carlo methods with the hybrid scheme, introduced in [8], the exact scheme from Bayer et al [4] and the difference Monte Carlo Scheme [20] for the direct approximation of the weak error.

2 Notation

In the master thesis the following notations are used.

- The Gamma function $\Gamma(x) : \mathbb{R}^+ \to \mathbb{R}$ is defined as

$$\Gamma(x) := \int_{0+}^{\infty} t^{(x-1)} e^{-t} dt.$$

  It holds that

$$\Gamma(n+1) = n!.$$

- ${}_2F_1$ stands for the Gauss hypergeometric function ${}_2F_1(a, b; c, z)$. If $|z| < 1$ then ${}_2F_1(a, b; c, z)$ is given by

$${}_2F_1(a, b; c; z) := \sum_{k=0}^{\infty} \frac{\Gamma(a+k)\Gamma(b+k)\Gamma(c)}{\Gamma(a)\Gamma(b)\Gamma(c+k)} \frac{z^k}{k!}.$$

- S_t is the value of an underlying at time t. S_t is a random variable.
- μ_t denotes the drift term of the random variable S_t at time t.
- W_t denotes a standard Brownian motion.
- $\mathbb{E}^{\mathbb{P}}[X] = \int_{\mathcal{Q}} X(\omega) d\mathbb{P}(\omega)$ denotes the expected value of a random variable X under the probability measure $\mathbb{P}$.
- $Var(X)$ is the variance of a random variable X. It is given by

$$Var(X) = \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2.$$

  If X has mean zero, it holds

$$Var(X) = \mathbb{E}\left[X^2\right].$$

- For two random variables X and Y the $Cov(X, Y)$ is defined as

$$Cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].$$

  If X or Y have mean zero it holds

$$Cov(X, Y) = \mathbb{E}[XY].$$

  Further it holds

$$Cov(X, X) = Var(X, X).$$

- ρ denotes the correlation coefficient of two random variables X_1 and X_2, which is defined as

$$\rho = \frac{Cov(X_1, X_2)}{\sqrt{Var(X_1)Var(X_2)}} \in [-1, 1]$$

- A function $f : \mathbb{R} \to \mathbb{R}$ is said to be ultimately monotone, if an $x_0 \in \mathbb{R}$ exists, such that f is monotone for all $x \geq x_0$. Each monotone function is also ultimately monotone.

- We write $f(x) = \mathcal{O}(h(x)),\ x \to a$, if

$$\limsup_{x \to a} \left| \frac{f(x)}{h(x)} \right| < \infty.$$

- By $L^p(\mathbb{R})$ we denote the space of all functions $f : \mathbb{R} \to \mathbb{C}$ with finite L^p-norm, i.e.

$$\|f\|_{L^p} = \left(\int_{\mathbb{R}} |f(x)^p| \, dx \right)^{\frac{1}{p}} < \infty,$$

  while $L^p_{bc}(\mathbb{R})$ denotes the space of function in $L^p(\mathbb{R})$ which are continuous and bounded.

- The convolution of two functions f and g $(f \star g)$ is defined as

$$(f \star g)(x) = \int_{\mathbb{R}} f(y)g(x - y)dy.$$

  The discrete convolution is given by

$$(f \star g)(x) = \sum_{y=-\infty}^{\infty} f(y)g(x - y).$$

- $\mathcal{E}$ is the *Wick* exponential. For a zero-mean Gaussian random variable Φ as

$$\mathcal{E}(\Phi) = \exp\left(\Phi - \frac{1}{2}\mathbb{E}\left[|\Phi|^2 \right] \right).$$

- $(X)^+$ denotes the maximum of X and 0, i.e. $(X)^+ = \max(0, X)$.

- $X_N \xrightarrow[N \to \infty]{L^2} X$ denotes the L^2 convergence, i.e.

$$\lim_{N \to \infty} \|X_N - X\|_{L_2} = 0.$$

- Hölder continuity: A function $f : \mathbb{R} \to \mathbb{R}$ is Hölder continuous or order α if a real C, $\alpha > 0$ exist, such that

$$|f(x) - f(y)| \leq C|x - y|^{\alpha}$$

  We write $f \in C^{\alpha}(\mathbb{R})$.

- Hölder's inequality: Let $p, q \in [1, \infty]$ with $\frac{1}{p} + \frac{1}{q} = 1$ and $f \in L^p$ and $ginL^q$. Then it holds that $fg \in L^1$ and

$$\|fg\|_{L^1} \leq \|f\|_{L^p} \|g\|_{L^q}.$$

- $cosh : \mathbb{R} \mapsto [1, \infty)$ denotes the cosinus hyperbolicus, given by

$$cosh(x) = \frac{1}{2}(e^x + e^{-x}),\ x \in \mathbb{R}.$$

3 Introduction to stochastic integration

In this chapter we aim to give a short instruction to stochastic calculus in form of stochastic differential equations and stochastic integration. First we look at the general form of stochastic differential equation and then we will introduce the Itô integral and the Riemann-Stieltjes.

In the following let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with normal filtration $(\mathcal{F}_t)_{t\geq 0}$ and probability measure $\mathbb{P}$.

3.1 Stochastic differential equations

In this thesis the models of the financial markets are given in form of stochastic differential equations (SDEs). Those have in general the form [23, chapter 6]

$$X_t = \mu_t dt + \sigma_t dW_t, \tag{3.1.1}$$

where $\mu : \mathbb{R} \to \mathbb{R}$ is the drift term of X_t, $\sigma_t : \mathbb{R} \to \mathbb{R}$ is a volatility process and W_t is a one dimensional Brownian motion, which is defined as given below [24, definition 11.2.2].

Definition 1 (Brownian motion) A Brownian motion is a real valued process $W = (W_t)_{t\in[0,T]}$ for which holds

- $W_0 = 0$ $\mathbb{P}$-a.s.
- W has independent increments
- $W_t - W_s \sim \mathcal{N}(0, t-s)$, $s \leq t \leq T$
- the paths $t \mapsto W_t$ are a.s. continuous.

If we write (3.1.1) in integral form we get

$$X_t = X_0 + \int_0^t \mu_s ds + \int_0^t \sigma_s dW_s.$$

The first integral is the classical Riemann integral but for the integral with respect to the Brownian motion we need another type of integral, the Itô integral, which we introduce in the following section.

3.2 Itô integral

In this section we briefly introduce the stochastic Itô-integral [cp. 22, 23 sction 3.2].

Definition 2 Let $H^2[0,T]$ be the vector space of all $\mathcal{F}$-adapted stochastic processes f, which fulfil

$$\mathbb{E}\left[\int_0^T f(\omega,t)^2 dt\right] < \infty.$$

Definition 3 $H_0^2[0,T]$ be the vector space of all piecewise constant functions in $H^2[0,T]$, i.e. $f \in H_0^2[0,T]$ if a partition $0 = t_0 < t_1 < \cdots < t_n = T$ exists, such that

$$f(\omega,t) = \sum_{i=0}^{n-1} a_i(\omega) \mathbb{1}_{\{t\in(t_i,t_{i+1}]\}}.$$

We call $f \in H_0^2[0,T]$ an elementary integrand.

For $f \in H_0^2[0,T]$ we can define the Itô-integral by

$$I(f) := \sum_{i=0}^{n-1} a_i (W_{t_{i+1}} - W_{t_i}).$$

We can further extend the definition of the Itô-integral to the whole $H^2[0,T]$ with the following lemma.

Lemma 1 *H_0^2 is dense in H^2, that means that for all $f \in H^2$ exist a sequence $f_n \in H_0^2$ such that*

$$\|f - f_n\|_{L_2} \xrightarrow{n\to\infty} 0.$$

For a proof see [39, chapter 6.6].

Therefore the Itô-integral for $f \in H^2[0,T]$ is given by the L^2 limes of $I(f_n)$, where f_n is the sequence in $H_0^2[0,T]$ for which holds $f_n \xrightarrow[L^2]{n\to\infty} f$.

For further analysis of the existence and uniqueness of the solution see for example [23, 3.2].

The Itô-integral fulfils the following properties.

Theorem 1 *For all $f \in H^2[0,T]$ it holds the following.*

- *Itô Isometry [23, lemma 3.4]*

$$\mathbb{E}\left[\left(\int_0^T f(w,t)dW_t\right)^2\right] = \mathbb{E}\left[\int_0^T f(w,t)^2 dt\right]$$

- *Expectation [13, prop. 4.13]*

$$\mathbb{E}\left[\int_0^T f(w,t)dW_t\right] = 0$$

- *If $f \in H^2[0,T]$ and has bounded total variation we have an integration by parts given by*

$$\int_0^T f(s)DW_s = f(t)W_t - \int_0^T W_s df(s).$$

If f is continuously differentiable we get

$$\int_0^T f(s)DW_s = f(t)W_t - \int_0^T W_s f'(s)ds.$$

This follows directly from the integration by parts property for general Itô integrals [19, proposition 7.5] and the properties of the Riemann-Stieltjes integral for continuously differentiable functions, which we introduce in the next section.

Remark 1 The Itô-integral can be further generalized to the group of semimartingales as integrands, see for example [19, chapter 5 & 7].
Note that the Brownian motion W is a martingale and thereby a semimartingale. Hence the Itô-integral with respect to W is a special case of the general Itô-integral for semimartingales.

3.3 Riemann-Stieltjes integral

In this section we shortly introduce the Riemann-Stieltjes integral. It can also be used for stochastic integration and we will later need it to define an integral with respect to the fractional Brownian motion W^H.
The Riemann-Stieltjes integral defines in general an integral for two functions f and g:

$$\int_a^b f(x)dg(x)$$

and is defined as follows [23, section 3.1]

Definition 4 (Riemann-Stieltjes integral) Let $f, g : [0, T] \to \mathbb{R}$ be two functions. $\mathcal{P}^n$ s sequence of partitions $0 = t_0^n < \cdots < t_N^n = T$ with $|\mathcal{P}^n| = \sup_i |t_{i+1}^n - t_i^n|$ and $|\mathcal{P}^n| \xrightarrow{n\to\infty} 0$. Further be $B^n : (b_i^n)_{i=1,\dots,N}$ a sequence with $b_i^n \in [t_i^n, t_{i+1}^n]$. If

$$\lim_{n\to\infty} \sum_{i=0}^{N-1} f(b_i^n) \left(g(t_{i+1}^n) - g(t_i^n) \right)$$

exists and is equal for all choices of $\mathcal{P}^n$ and B^n we call this the *Riemann-Stieltjes integral*

$$\int_0^T f(x) dg(x)$$

of f with respect to g.

We write $f \in R(g)$ if f is Riemann-Stieltjes integrable with respect to g.

Remark 2 The Riemann-Stieltjes doesn't exist for all functions f and g, but an well know result states that for continuous f and g of bounded total variation, i.e.
$\sup_{\mathcal{P}^n} \sum_{i=0}^{N-1} |g(t_{i+1}^n) - g(t_i^n)| < \infty$, the Riemann-Stieltjes integral

$$\int_0^T f(x) dg(x)$$

exists [32, chapter 8].

In the following we present some helpful properties of the Riemann–Stieltjes integral [19, prop. 2.4 & 2.5], [32, chapter 8]

- linearity in the integrand: for $c \in \mathbb{R}$ and $f, h \in R(g)$ it holds

$$\int_a^b cf(x) + h(x) dg(x) = c \int_a^b f(x) dg(x) + \int_a^b h(x) dg(x)$$

- linearity in the integrator: for $c \in \mathbb{R}$ and $f \in R(g)$ and $f \in R(h)$ it holds

$$\int_a^b f(x) dcg(x) + h(x) = c \int_a^b f(x) dg(x) + \int_a^b f(x) dh(x)$$

- for $c \in \mathbb{R}$ it holds

$$\int_a^b f(x) d(g(x) + c) = \int_a^b f(x) dg(x)$$

- if $f \in R(g)$ and g is continuously differentiable, then it holds

$$\int_a^b f(x)dg(x) = \int_a^b f(x)g'(x)dx.$$

Similar to the Itô integral with respect to the Brownian motion W we have an integration by parts property for the Riemann-Stieltjes integral [32, chapter 8].

Theorem 2 *If $f \in R(g)$ on $[s,t]$ or $g \in R(f)$ on $[s,t]$ it holds*

$$\int_s^t f(x)dg(x) = f(t)g(t) - f(s)g(s) - \int_s^t g(x)df(x).$$

Remark 3 The Riemann-Stieltjes integral can't be used for the Brownian motion, since W has unbounded total variation. Therefore exists a function f for that the Riemann-Stieltjes integral isn't defined [23, lemma 3.1., theorem 3.2.]. Thus we needed the Itô integral for the integration with respect to W.

4 Rough volatility models

Rough volatility models are stochastic volatility models where the volatility process v_t is described by a fractional Brownian motion with Hurst parameter $H < \frac{1}{2}$, see section 4.1 for an introduction. In stochastic volatility models in general the volatility process v_t is given by a stochastic differential equation. The following models are examples of classical stochastic volatility models [6].

- the Heston model:

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{v_t} S_t dW_t^1 \\ dv_t &= \kappa(\theta - v_t)dt + \xi\sqrt{v_t}\left(\rho dW_t^1 + \sqrt{1-\rho^2}dW_t^2\right), \end{aligned}$$

  where κ, θ, $\xi > 0$ are parameters and $\rho \in [-1, 1]$ is the correlation between W_1 and W_2

- the Stochastic Alpha Beta Rho (SABER) model:

$$\begin{aligned} dS_t &= v_t S_t^{\beta} dW_t^1 \\ dv_t &= \alpha v_t\left(\rho dW_t^1 + \sqrt{1-\rho^2}dW_t^2\right), \end{aligned}$$

  where $\alpha \geq 0$, $0 \leq \beta \leq 1$ and $\rho \in [-1, 1]$

- the Stein-Stein model:

$$\begin{aligned} dS_t &= \mu S_t dt + |v_t| S_t dW_t^1 \\ dv_t &= \kappa(\theta - v_t)dt + \eta dW_t^2, \end{aligned}$$

  with $\kappa, \eta > 0$ and W_t^1 and W_t^2 are correlated with correlation coefficient ρ.

The first rough volatility model, the Rough Stochastic Volatility model, was introduced by Gatheral et al in [15]. Recently Bayer et al developed an other rough volatility model called the rough Bergomi model [4].
Both models developed from an empirical study by Gatheral et al [15], where they found that the time series of realized variance of various assets is consistent with the simple model

$$\log(\sigma_{t+\Delta}) - \log(\sigma_t) = \nu(W_{t+\Delta}^H - W_t^H), \tag{4.0.1}$$

where W_t^H is a fractional Brownian motion with Hurst parameter $H < \frac{1}{2}$ and $\nu > 0$ is a positive constant. (4.0.1) may also be written as

$$\sigma_t = \sigma \exp(\nu W_t^H), \tag{4.0.2}$$

with $\sigma > 0$.
Before we introduce the rough volatility models we will give an introduction to the fractional Brownian motion. After that we introduce the Rough Fractional Stochastic Volatility model by Gatheral et al [15] and the rough Bergomi model by Bayer et al [4].

4.1 Fractional Brownian motion

A fractional Brownian motion (fBm) is a modification of the classical Brownian motion. It has a different covariance than the classical Brownian motion, which depends on the so called Hurst parameter $H \in (0, 1)$ [30 , Def. 1.2.1].

Definition 5 (Fractional Brownian motion) The (two-sided, normalized) *fractional Brownian motion* (fBm) with Hurst parameter $H \in (0, 1)$ is a Gaussian process $W^H = \{W_t^H, t \in \mathbb{R}\}$ on $(\Omega, \mathcal{F}, \mathbb{P})$ having the properties

1. $W_0^H = 0$
2. $\mathbb{E}[W_t^H] = 0$, $t \in \mathbb{R}$
3. $\mathbb{E}[W_t^H W_s^H] = \frac{1}{2}(t^{2H} + s^{2H} - |t - s|^{2H})$, $s, t \in \mathbb{R}$.

Remark 4 It is possible to consider the fBm only on $\mathbb{R}_+$, which is then called one-sided fBm [30, Remark 1.2.4].

The fBm fulfils the following properties [34, p. 1542 - 1543]:

- The fBm is self stationary. For any constant $a > 0$, the processes $\{a^{-H} W_{at}^H, t \geq 0\}$ and $\{W_t^H, t \geq 0\}$ have the same probability distribution.
- The increments are normal distributed with mean zero and variance

$$\mathbb{E}[(W_t^H - W_s^H)^2] = |t - s|^{2H}.$$

- For all $\varepsilon > 0$ and $T > 0$, there exists a non-negative random variable $X(\varepsilon, T)$ such that $\mathbb{E}[|X(\varepsilon, T)|^p) < \infty$ for all $p \geq 1$, and almost sure

$$|W_t^H - W_s^H| \leq X(\varepsilon, T)|t - s|^{H-\varepsilon} \ \forall s, t \in [0, T].$$

  Hence the sample parts are Hölder continuous of order $H - \varepsilon$, for any $\varepsilon > 0$.

Remark 5 A classical Brownian motion has the same properties 1 and 2 as a fBm, see definition 5, but instead for the covariance holds

$$\mathbb{E}[W_t W_s] = \min(t, s).$$

The fBm is a generalization of the classical Brownian motion. For $H = \frac{1}{2}$ we obtain the normal Brownian motion. W.l.o.g. be $t \leq s$, then it holds

$$Cov(W_t^{\frac{1}{2}}, W_s^{\frac{1}{2}}) = \frac{1}{2}(t + s - |t - s|) = \frac{1}{2} * 2t = t$$

thus

$$Cov(W_t^{\frac{1}{2}}, W_s^{\frac{1}{2}}) = \min(t, s).$$

Hence $W^{\frac{1}{2}}$ fulfils the properties of the Brownian motion.

Remark 6 The sample parts of a normal Brownian motion are $\left(\frac{1}{2} - \varepsilon\right)$-Hölder continuous but the fBm sample parts are $(H - \varepsilon)$-Hölder continuous. They are "rougher" than the normal Brownian motion for $H < \frac{1}{2}$. Hence the models are called rough volatility models.

Mandelbrot and Van Ness introduced a representation of the fBm in terms of integrals with respect to Brownian motions [28, Def. 2.1], the so called *Mandelbrot - Van Ness representation of fBm*.

Remark 7 The Mandelbrot - Van Ness representation is given by

$$W_t^H = C_H \left\{ \int_{-\infty}^{t} \frac{1}{(t-s)^\gamma} dW_s - \int_{-\infty}^{0} \frac{1}{(-s)^\gamma} dW_s \right\}, \tag{4.1.1}$$

where the choice $\gamma = \frac{1}{2} - H$ and

$$C_H = \sqrt{\frac{2H\Gamma(3/2 - H)}{\Gamma(H + 1/2)\Gamma(2 - 2H)}}.$$

ensures that $\mathbb{E}[W_t^H W_s^H] = \frac{1}{2}(t^{2H} + s^{2H} - |t - s|^{2H})$.

In the rBergomi model, see section 4.4, by Bayer et al [4], the fBm appears in form of the Mandelbrot-Van Ness representation.

Unfortunately the fBm W^H is not a semimartingale for $H \neq \frac{1}{2}$ [41, section 2.3]. Therefore we can't use Itô calculus for the integration with respect to the Brownian motion W^H. Instead we need to introduce a different type of integral, which allows us to integrate with respect to W^H for $H \neq \frac{1}{2}$.

4.2 Integration with respect to fBm

For the integration with respect to a fBm W^H for $H \neq \frac{1}{2}$ we will use the Young integral as in [33]. For an overview on the topic of integration with respect to the fBm see for example [41]. In the following be $H \neq \frac{1}{2}$.

In [43] it is shown, that for certain Hölder continuous functions f, g the Riemann-Stieltjes integral exists, if f and g are smooth enough.

Theorem 3 *Let $f \in C^\alpha([0,T])$ and $g \in C^\alpha([0,T])$. If $\alpha + \beta > 1$, then for any $[s,t] \subset [0,T]$ the integral*

$$\int_s^t f(x)dg(x)$$

exists as a Riemann-Stieltjes integral.

The theorem is based on a result by Young [42], who extended the use of Riemann-Stieltjes integrals. Therefore the integral is called *Young integral*. So following the theorem 3, the *Young integral* $\int_0^T f(t)dg(t)$ is defined as the limit of the corresponding Riemann-Stieltjes sums.

Therefore the pathwise Riemann-Stieltjes integral

$$\int_s^t f(u)dW_u^H, \ -T \leq s < t \leq T$$

exists if $f \in C^\alpha([-T,T])$ with $\alpha + H - \varepsilon > 1$ due to the Hölder smoothness of the fBm W^H.

For $-T \leq s_1 < t_1 \leq s_2 < t_2 < T$ the fractional Itô isometry holds

$$\mathbb{E}\left[\int_{s_1}^{t_1} f(u_1)dW_{u_1}^H \int_{s_2}^{t_2} f(u_2)dW_{u_2}^H\right] = H(2H-1)\int_{s_1}^{t_1}\int_{s_2}^{t_2} f(u_1)f(u_2)|u_1-u_2|^{2H-2}du_2du_1. \tag{4.2.1}$$

For the proof of the strong error, see section 6.1, we will need the following integral with respect to the fBm W^H

$$\int_{-\infty}^t e^{\lambda s}dW_s^H(\omega), \ t \in \mathbb{R}, \ \omega \in \Omega. \tag{4.2.2}$$

In [10, proposition A.1] the existence of the improper Riemann-Stieltjes integral (4.2.2) is shown. Further it holds

$$\int_{-\infty}^t e^{\lambda s}dW_s^H(\omega) = e^{\lambda t}W_t^H(\omega) - \lambda\int_{-\infty}^t e^{\lambda u}W_u^H(\omega)du, \ t \in \mathbb{R}, \ \omega \in \Omega$$

which follows directly from the integration by parts theorem 2 for the Riemann-Stieltjes integral.

Also the isometry (4.2.1) is still valid [10, lemma 2.3]

$$\int_{-\infty}^{t_1} e^{\lambda x_1} dW_{x_1}^H \int_{t_2}^{t_3} e^{\lambda x_2} dW_{x_2}^h = H(2H-1) \int_{-\infty}^{t_1} \int_{t_2}^{t_3} e^{\lambda(x_1+x_2)} |x_1 - x_2|^{2H-2} dx_2 dx_1$$

for $-\infty < t_1 \leq t_2 \leq t_3 < \infty$.

4.3 Rough Fractional Stochastic Volatility model

The Rough Stochastic Volatility (RFSV) model was developed by Gatheral et al [15] and is defined as follows

$$\frac{dS_t}{S_t} = \mu_t dt + \sigma_t dZ_t \tag{4.3.1}$$

$$\sigma_t = \exp(Y_t), \ t \in [0, T], \tag{4.3.2}$$

where μ_t is a drift term and Z_t is a standard Brownian motion. Y_t is a Ornstein-Uhlenbeck process driven by a fBm, which is called a fractional Ornstein-Uhlenbeck process, given by

$$dY_t = \theta dW_t^H - \lambda(Y_t - \nu)dt, \tag{4.3.3}$$

where $\nu \in \mathbb{R}$, θ, $\lambda > 0$ and W_t^H is a fBm with Hurst parameter H.

Gatherhal et al used empirical result (4.0.2) to derive the volatility process Y_t. Since (4.0.2) is not stationary and stationary was a desired property, they modelled the log-volatility by the stationary Ornstein-Uhlenbeck process (Y_t) (4.3.3). If we set $\lambda = 0$ in (4.3.3), we obtain again (4.0.2).

Gatheral et al [15] found that the RFSV model fits empirical market data remarkably well. Further they have done a comparison of the simulated and the S&P volatility for 3,500 days and found that the model creates a graph with the same property as the S&P volatility [15, Figure 3.6].

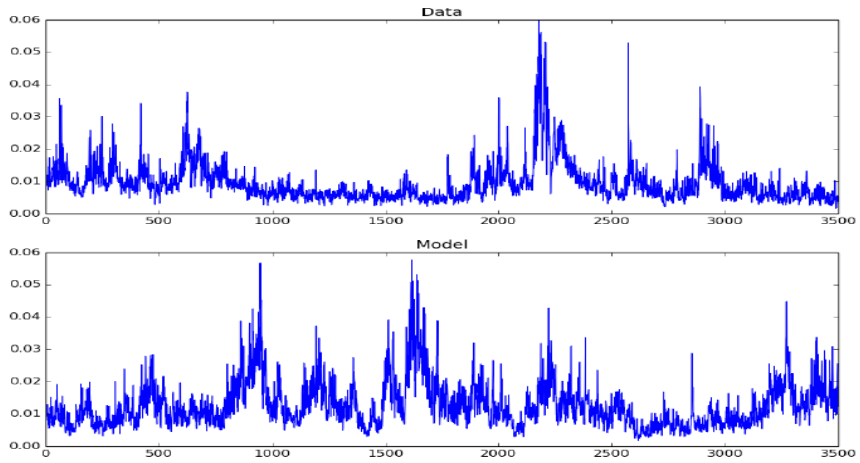

Figure 2: Volatility of the S&P (Data) and of the RFSV model (Model).

Before we continue with the introduction of the rough Bergomi model, we present some further properties of Y_t [33, p. 6], which we later need for the proof of the strong error in section 6.1.

$$dY_t = \theta dW_t^H - \lambda(Y_t - \nu)dt = \lambda(\nu - Y_t)dt + \theta dW_t^H$$

has a stationary solution which is given by [10]

$$Y_t = \nu + \theta e^{-\lambda t} \int_{-\infty}^{t} e^{\lambda u} dW_u^H, \ t \in \mathbb{R}.$$

The stationarity is a consequence of the shift invariance of the fBm, see [33, p. 6].

Furthermore it holds that Y is Gaussian with

$$\mathbb{E}[Y_t] = 0$$

and

$$Cov(Y_t, Y_s) = R_Y(|t - s|), \ s, t \in \mathbb{R},$$

where

$$R_Y(\tau) = \theta^2 \left(\frac{\Gamma(2H+1)\cosh(\lambda\tau)}{2\lambda^{2H}} - H \int_0^\tau \cosh(\lambda(\tau - u))u^{2H-1} \right), \ \tau \geq 0. \tag{4.3.4}$$

For the variance it holds

$$Var(Y_t) = R_Y(0) = \theta^2 \frac{\Gamma(2H+1)}{2\lambda^{2H}},\ t \in \mathbb{R}.$$

For a derivation of $Var(Y_t)$ see the Appendix of [33]. Further we need for the proof of the strong error the following lemma [33, lemma 2].

Lemma 2 *We have* $R_Y \in C^\infty((0,\infty);\mathbb{R})$ *and*

$$\begin{aligned} R_Y(\tau) &= R_Y(0) - \frac{\theta^2}{2}\tau^{2H} + o(\tau^{2H}) \\ &= Var(Y_t) - \frac{\theta^2}{2}\tau^{2H} + o(\tau^{2H}),\ \tau \to 0. \end{aligned}$$

4.4 Rough Bergomi model

The rough Bergomi model (further rBergomi model) was developed by C. Bayer et al [4]. The model is defined as

$$\frac{dS_t}{S_t} = \sqrt{v_t} dZ_t,$$

with volatility process

$$v_u = \mathbb{E}^{\mathbb{P}}[v_u|\mathcal{F}_t]\varepsilon\left(\eta\tilde{W}_t^{\mathbb{Q}}(u)\right)\exp\left\{\eta\sqrt{2H}\int_t^u \frac{1}{(u-s)^\gamma}\lambda(s)ds\right\} \tag{4.4.1}$$

$$= \xi_t(u)\mathcal{E}\left(\eta\tilde{W}_t^{\mathbb{Q}}(u)\right) \tag{4.4.2}$$

and

$$dZ_t = \rho dW_t + \sqrt{1-\rho^2}dW_t^\perp,$$

where ρ is the correlation between the Brownian motions W, which is driving the volatility moves, and $W^\perp$, which is driving the price moves. $\mathbb{Q}$ is the equivalent martingale measure $\mathbb{P} \sim \mathbb{Q}$ on $[t,T]$ such that S_t is a martingale under $\mathbb{Q}$, which we need to get an arbitrage free price due to the fundamental theorem of asset pricing [22]. Further $\tilde{W}_t^{\mathbb{Q}}$ is defined as

$$\tilde{W}_t^{\mathbb{Q}}(u) := \sqrt{2H}\int_t^u \frac{1}{(u-s)^\gamma} dW_s^{\mathbb{Q}} \tag{4.4.3}$$

and

$$\xi_t(u) = \mathbb{E}^{\mathbb{P}}[v_u|\mathcal{F}_t]\exp\left\{\eta\sqrt{2H}\int_t^u \frac{1}{(u-s)^\gamma}\lambda(s)ds\right\}.$$

The rBergomi model is non-Markovian in the instantaneous variance v_t : $\mathbb{E}^{\mathbb{Q}}[v_u|\mathcal{F}_t] \neq \mathbb{E}^{\mathbb{Q}}[v_u|v_t]$, see [4, p.9]. But it is Markovian in the state vector by definition $\mathbb{E}^{\mathbb{Q}}[v_u|\mathcal{F}_t] = \xi_t(u)$.

Remark 8 The model is a non Markovian version of the Bergomi model [9]. In the classical n-factor Bergomi model [9] the variance curve model may be written as

$$\xi_t(u) = \xi_0(u)\varepsilon\left(\sum_{i=1}^{n}\eta_i\int_0^t e^{-\kappa_i(u-s)}dW_s^{(i)}\right).$$

Hence the model is called rough Bergomi model.

Note that $\tilde{W}$ (4.4.3) is a so called *Riemann-Liouville fractional Brownian motion*, which got introduced by Sithi and Lim [38]. Besides it is a Brownian semi-stationary process (further called BSS), which were introduced by Barndorff-Nielsen and Schmiegel [2, 3].

Definition 6 (Brownian semi-stationary process) X_t is called a *Brownian semi-stationary process* if

$$X_t = \int_{-\infty}^{t} g(t-s)\sigma_s dW_s \tag{4.4.4}$$

for some deterministic kernel function g and an adapted intermittency process σ. If the integral starts at 0 instead of $-\infty$

$$X_t = \int_{0}^{t} g(t-s)\sigma_s dW_s \tag{4.4.5}$$

we call the process *truncated Brownian semi-stationary process* (TBSS).

4.4.1 Derivation of the rBergomi model

We will shortly repeat the derivation of the rBergomi model [4, Section 2]. For the derivation we will use (4.0.1) and the Mandelbrot - Van Ness representation (4.1.1) of the fBm.

Using the terms $v_t = \sigma_t^2$ and using (4.1.1) in (4.0.1) we get

$$\begin{aligned}
&\log(v_u) - log(v_t) = 2(\log(\sigma_u)) - \log(\sigma_t)) = 2\nu(W_u^H - W_t^H)\\
&= 2\nu C_H\left(\int_{-\infty}^{u}\frac{1}{(u-s)^\gamma}dW_s^{\mathbb{P}} - \int_{-\infty}^{0}\frac{1}{(-s)^\gamma}dW_s^{\mathbb{P}} - \int_{-\infty}^{t}\frac{1}{(t-s)^\gamma}dW_s^{\mathbb{P}} + \int_{-\infty}^{0}\frac{1}{(-s)^\gamma}dW_s^{\mathbb{P}}\right)\\
&= 2\nu C_H\left(\int_{-\infty}^{u}\frac{1}{(u-s)^\gamma}dW_s^{\mathbb{P}} - \int_{-\infty}^{t}\frac{1}{(t-s)^\gamma}dW_s^{\mathbb{P}}\right)\\
&= 2\nu C_H\left(\int_{t}^{u}\frac{1}{(u-s)^\gamma}dW_s^{\mathbb{P}} + \int_{-\infty}^{t}\frac{1}{(u-s)^\gamma} - \frac{1}{(t-s)^\gamma}dW_s^{\mathbb{P}}\right)\\
&=: 2\nu C_H(M_t(u) + Z_t(u)),
\end{aligned}$$

where $Z(t)$ is $\mathcal{F}_t$ measurable and $M_t(u)$ is independent of $\mathcal{F}_t$ and Gaussian with mean zero and variance $\frac{(u-t)^{2H}}{2H}$. Thereby we get a representation of v_u:

$$v_u = v_t \exp(\log(v_u) - \log(v_t)) = v_t \exp(2\nu C_H(M_t(u) + Z_t(u)).$$

With the definition 4.4.3 of $\tilde{W}_t^{\mathbb{P}}$ and $\eta = \frac{2\nu C_h}{\sqrt{2H}}$ we have $2\nu C_H M_t(u) = \eta \tilde{W}_t^{\mathbb{P}}(u)$. Note that $\tilde{W}_t^{\mathbb{P}}$ has the same properties as $M_t(u)$, only that the variance is $(u-t)^{2H}$. Therefore by replacing $2\nu C_H M_t(u)$ we get

$$\begin{aligned} v_u &= v_t \exp(2\nu C_h(M_t(u) + Z_t(u)) \\ &= v_t \exp(2\nu C_H Z_t(u) + \eta \tilde{W}_t^{\mathbb{P}}(u)). \end{aligned}$$

Thus we get the following model under the physical probability measure $\mathbb{P}$

$$\begin{aligned} \frac{dS_u}{S_u} &= \mu_u d_u + \sqrt{v_u} dZ_u^{\mathbb{P}} \\ v_u &= v_t \exp\left(\eta \tilde{W}_t^{\mathbb{P}}(u) + 2\nu C_H Z_t(u)\right), \end{aligned}$$

where $Z^{\mathbb{P}}$ and $W^{\mathbb{P}}$ are two standard Brownian motions with correlation ρ.

Note we can also write v_u in terms of the Wick stochastic exponential $\mathcal{E}(\cdot)$. Therefore we need

$$\mathbb{E}[v_u|\mathcal{F}_t] = v_t \exp\left(2\nu C_H Z_t(u) + \frac{1}{2}\eta^2 \mathbb{E}[\tilde{W}_t^{\mathbb{P}}(u)]^2\right).$$

Then we get

$$v_u = v_t \exp\left(\eta \tilde{W}_t^{\mathbb{P}}(u) + 2\nu C_H Z_t(u)\right) \tag{4.4.6}$$

$$= v_t \exp\left(2\nu C_H Z_t(u) + \frac{1}{2}\eta^2 \mathbb{E}[\tilde{W}_t^{\mathbb{P}}(u)]^2\right) \exp\left(\eta \tilde{W}_t^{\mathbb{P}}(u) - \frac{1}{2}\eta^2 \mathbb{E}[\tilde{W}_t^{\mathbb{P}}(u)]^2\right) \tag{4.4.7}$$

$$= \mathbb{E}^{\mathbb{P}}[v_u|\mathcal{F}_t]\mathcal{E}\left(\eta \tilde{W}_t^{\mathbb{P}}\right). \tag{4.4.8}$$

For getting the under section 4.4 stated rBergomi model for the pricing of the underlying S_t we still need to change the physical measure $\mathbb{P}$ to the equivalent martingale measure $\mathbb{Q}$. For that consider the measure change obtained by a Girsanov change of measure, see for example [23, section 7] [22],

$$dZ_u^{\mathbb{Q}} = dZ_u^{\mathbb{P}} + \frac{\mu_u}{\sqrt{v_u}} du, \ t \leq u \leq T$$

on a fixed time interval $[t, T]$. Thus we get

$$\frac{dS_u}{S_u} = \sqrt{v_u} dZ_u^{\mathbb{Q}}, \ t \leq u \leq T.$$

$\tilde{W}_t^{\mathbb{P}}$ has been built from $W^{\mathbb{P}}$, which is correlated with $Z_u^{\mathbb{P}}$, such that

$$dW_u^{\mathbb{P}} = \rho dZ_u^{\mathbb{P}} + \bar{\rho} d\bar{Z}_u^{\mathbb{P}},\ \rho^2 + \bar{\rho}^2 = 1,$$

where $(Z^{\mathbb{P}}, \bar{Z}^{\mathbb{P}})$ are independent standard Brownian motions.
If we consider the following change of measure for $\bar{Z}_u^{\mathbb{P}}$

$$d\bar{Z}_u^{\mathbb{Q}} = d\bar{Z}_u^{\mathbb{P}} + \gamma_u du,$$

where $\gamma = \gamma(u, \omega)$, for $u \in [t, T]$, is a suitable adapted process called the *market price of volatility risk*, we get

$$\begin{aligned} dW_u^{\mathbb{Q}} &= \rho dZ_u^{\mathbb{Q}} + \bar{\rho} dZ_u^{\mathbb{Q}} \\ &= \rho \left(dZ_u^{\mathbb{P}} + \frac{\mu_u}{\sqrt{v_u}} \right) + \bar{\rho} \left(dZ_u^{\mathbb{P}} + \gamma_u du \right) \\ &= dW_u^{\mathbb{P}} + \left(\rho \frac{\mu_u}{\sqrt{v_u}} + \bar{\rho} \gamma_u \right) du,\ t \leq u \leq T, \end{aligned}$$

which we may rewrite as

$$dW_s^{\mathbb{Q}} = dW_s^{\mathbb{P}} + \lambda_s ds. \tag{4.4.9}$$

Now we are able to carry out the change of measure in v_u. If we use (4.4.9) in (4.4.8) we receive

$$\begin{aligned} v_u &= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \mathcal{E}\left(\eta \tilde{W}_t^{\mathbb{P}} \right) \\ &= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \exp\left(\eta \sqrt{2H} \int_t^u \frac{1}{(u-s)^{\gamma}} dW_s^{\mathbb{P}} - \frac{\eta^2}{2} (u-t)^{2H} \right) \\ &= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \mathcal{E}\left(\eta \tilde{W}_t^{\mathbb{Q}} \right) exp\left(\eta \sqrt{2H} \int_t^u \frac{\lambda_s}{(u-s)^{\gamma}} ds \right). \end{aligned}$$

Hence we get the form of the rBergomi model introduced in section 4.4.

5 Numerical Simulation

In this section we are going to look at the numerical simulation methods we need for simulating a financial model like the rBergomi model.
At first we are going to repeat the discretization of stochastic differential equations and the approximation errors we get by the discretization [6, chapter 3]. Then we will present the classical Monte Carlo method [6, chapter 2.2] and finally introduce the methods we will use for the identification of the weak error and the rate of convergence of the discretization methods.

5.1 Discretization of stochastic differential equations

Let us consider a general n-dimensional stochastic differential equation driven by a d-dimensional Brownian motion W, i.e.,

$$\begin{aligned} dX_t &= a(X_t)dt + b(X_t)dW_t \\ X_0 &= x, \end{aligned}$$

where a and b are functions from $\mathbb{R}^d \mapsto \mathbb{R}$ and $x \in \mathbb{R}^d$. We want to determine an approximation $\hat{X}$ on a fixed time interval $[0, T]$ for X.

For a fixed time interval $[0, T]$ the approximation is based on a time grid $D = \{0 = t_0 < t_1 < \cdots < t_N = T\}$ with size N, where N is the number of discretization steps. We denote by

$$|D| := \max_{1 \leq i \leq N} |t_i - t_{i-1}|$$

the mesh of the grid. If we have a homogeneous grid it holds for given N, that

$$|D| = \frac{1}{N}. \tag{5.1.1}$$

The best known discretization scheme is the *Euler-Maruyama scheme* [16, p. 340], which is given by

$$\begin{aligned} \hat{X}_{t_0} &= x \\ \hat{X}_{t_{i+1}} &= \hat{X}_{t_i} + a(\hat{X}_{t_i})(t_{i+1} - t_i) + b(\hat{X}_{t_i})(W_{t_{i+1}} - W_{t_i}), \end{aligned}$$

where $i = 0, \ldots, N-1$.
Another discretization scheme is the *Euler-trapezoidal scheme*

$$\begin{aligned} \hat{X}_{t_0} &= x \\ \hat{X}_{t_{i+1}} &= \hat{X}_{t_i} + a\left(\frac{1}{2}(\hat{X}_{t_{i+1}} + \hat{X}_{t_i})\right)(t_{i+1} - t_i) + b\left(\frac{1}{2}(\hat{X}_{t_{i+1}} + \hat{X}_{t_i})\right)(W_{t_{i+1}} - W_{t_i}). \end{aligned}$$

It follows directly from the Euler-Maruyama scheme by applying the trapezoidal rule

$$\int_{t_i}^{t_{i+1}} f(s)ds \approx \frac{1}{2}(f(t_{i+1}) + f(t_i))(t_{i+1} - t_i).$$

5.2 Approximation error

An important part of numerical calculations is the error analysis. Since we only calculate an approximation for the SDE it is important to verify how good a given approximation is. Indeed we would like to be able to control the error, i.e., we would like to have an error estimate and we would like to know how fast the error goes to zero if we increase N. [cp. 6, Chapter 3.1].

5.2.1 Strong Error

The definition is from [6, definition 3.1].

Definition 7 (Strong approximation error) The scheme $\hat{X}^D$ *converges strongly* to X if

$$\lim_{|D|\to 0} \mathbb{E}[|X_T - \hat{X}_T^D|] = 0.$$

We say a scheme $\hat{X}^D$ has strong order of γ if for $|D|$ small enough

$$\mathbb{E}[|X_T - \hat{X}_T^D|] \leq C|D|^\gamma$$

for some constant $C > 0$, which does not depend on $\gamma > 0$.

Remark 9 The *strong approximation error* also is often defined in terms of the L_p norm, see for example [25, 40],

$$\|X_T - \hat{X}_T^D\|_{L_p(\Omega,H)} \leq C|D|^\gamma,$$

where $\|\cdot\|_{L_p(\Omega,H)}$ is given by

$$\|X\|_{L_p(\Omega,H)} = \mathbb{E}[X^p]^{\frac{1}{p}}.$$

A special form of this error is the *root mean squared error*, which we obtain for $p = 2$

$$\mathbb{E}[(X_T - \hat{X}_T^D)^2]^{\frac{1}{2}}.$$

5.2.2 Weak Error

The definition is from [6, definition 3.2].

Definition 8 (Weak approximation error) Given a suitable class $\mathcal{G}$ of functions $f : \mathbb{R}^n \to \mathbb{R}$, we say that scheme $\hat{X}^D$ *converges weakly* with respect to $\mathcal{G}$ if

$$\forall f \in \mathcal{G} : \lim_{|D|\to 0} \left|\mathbb{E}[f(\hat{X}_T^D) - \mathbb{E}[f(X_T)]\right| = 0.$$

We say that $\hat{X}^D$ has weak order $\gamma > 0$ if for every $f \in \mathcal{G}$ there is a constant $C > 0$ (not depending on $|D|$) such that

$$\left|\mathbb{E}[f(\hat{X}_T^D)] - \mathbb{E}[f(X_T)]\right| \leq C|D|^\gamma$$

provided that $|D|$ is small enough.

Remark 10 We will just work with homogeneous grids so 5.1.1 always holds. Therefore we use 5.1.1 to write the weak order as

$$\left|\mathbb{E}[f(\hat{X}_T^N)] - \mathbb{E}[f(X_T)]\right| \leq CN^{-\gamma}.$$

Hence we will give our simulation results in terms of the total number of discretization steps N.

In computational finance the weak error is often more interesting than the strong error because most approximation problems in finance are of the weak type for example derivative pricing.

5.3 Monte Carlo simulation

So far we have introduced the approximation of a stochastic differential equation but our goal is to get a solution for

$$\mathbb{E}[f(X)],$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a real function, X is a given random variable and $f(X)$ is integrable. In our case X will be given by the rBergomi model. We can get an approximated solution for $\mathbb{E}[f(X)]$ by using the Monte Carlo estimator [6, p. 12].

Definition 9 (Monte Carlo estimator) Be $N \in \mathbb{M}$, $X_1, X_2, \ldots, X_M$ independent realisations of X and $f : \mathbb{R}^d \to \mathbb{R}$ integrable. Than the Monte Carlo estimator for $\mathbb{E}[f(X)]$ is given by

$$\frac{1}{M}\sum_{i=1}^{M} f(X_i).$$

The Monte Carlo estimator indeed convergences to $\mathbb{E}[f(X)]$ because with the law of large numbers [24, corollary 10.2.22] it holds

$$\mathbb{E}[f(X)] = \lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} f(X_i),\ \mathbb{P}-a.s.$$

Since the Monte Carlo estimator is only an estimator for $\mathbb{E}[f(X)]$, we also need to consider the error we get. In case of the Monte Carlo estimator we don't call the error approximation error, since we don't approximate a stochastic differential equation. Instead we call him integration error, since we approximate $\mathbb{E}[\,\cdot\,]$, which is a stochastic integral.

Definition 10 (Monte Carlo integration error) Be $M \in \mathbb{N}$, X_M i.i.d. realizations of X and $f : \mathbb{R}^d \to \mathbb{R}$ integrable. The Monte Carlo integration error is given by

$$\left|\mathbb{E}[f(X)] - \frac{1}{M}\sum_{i=1}^{M} f(X_i)\right|.$$

To look at the behaviour of the integration error, we look at the root mean square error of the Monte Carlo estimator.

Theorem 4 (6, Proposition 2.15) *Let $X_1, \dots, X_M$ be i.i.d. realisations of X and $f : \mathbb{R}^d \to \mathbb{R}$ integrable. Then*

$$\mathbb{E}\left[\left(\mathbb{E}[f(X)] - \frac{1}{M}\sum_{i=1}^{M} f(X_i)\right)^2\right]^{\frac{1}{2}} = \sqrt{\frac{Var(f(X))}{M}},$$

for all $M \in \mathbb{N}$.

Proof Since $(X_i)_{i=1}^M$ are i.i.d. it holds with the formula for the variance of a sum [24, lemma 3.5.2]

$$\begin{aligned} Var\left(\sum_{i=0}^{M} f(X_i)\right) &= \sum_{i=1}^{M} Var(f(X_i)) + \sum_{\substack{i,j=1\\ i\neq j}}^{M} Cov(X_i, X_j) \\ &= \sum_{i=1}^{M} Var(f(X_i)), \end{aligned}$$

since independent random variables are uncorrelated.

Thus it holds

$$\begin{aligned}\mathbb{E}\left[\left(\mathbb{E}[f(X)] - \frac{1}{M}\sum_{i=1}^{M} f(X_i)\right)^2\right] &= Var\left(\frac{1}{M}\sum_{i=0}^{M} f(X_i)\right) \\ &= \frac{1}{M^2}\sum_{i=1}^{M} Var\left(f(X_i)\right) \\ &= \frac{MVar\left(f(X_1)\right)}{M^2} = \frac{Var\left(f(X_1)\right)}{M}.\end{aligned}$$

Taking the square root gives the desired result. □

We say, conclude that the typical error, i.e. the root mean square error, decreases to zero like $\frac{1}{\sqrt{M}} = M^{-\frac{1}{2}}$. Hence we say that the Monte Carlo method converges with rate $\frac{1}{2}$ respectively has order of convergence $\frac{1}{2}$.
Furthermore the integration error depends on the variance of X. When we increase our number of discretization steps N in our approximation the variance of $\hat{X}_N$ increases. Hence we get a higher integration error if we increase N.

5.4 Testing of Monte Carlo methods

In this section we will take a look at different methods to determine the weak error and the order of convergence of the weak error.
Because no closed form solution of the rBergomi model exists, we only can use Monte Carlo simulations to determine a solution. Hence one method we can use to determine a weak error is to simulate a solution of the rBergomi model for a high number of discretization steps and Monte Carlo paths and use this solution as a reference solution calculate the weak error with this approximated reference solutions. Thus we obtain a weak error for the rBergomi model.
Once we have a weak error we can calculate the rate of convergence and the experimental order of convergence, which we will describe in section 5.4.1.
Further we will try a different approach that can be used to examine the convergence of Monte Carlo simulations where no comparison solution exists: the difference Monte Carlo scheme, which we introduce in section 5.4.2.

In section 9 we will give our results for the exact scheme and in section 10 for the hybrid scheme.

Remark 11 Note that in our simulations we always calculate the *computational error* of the Monte Carlo simulation

$$\left|\mathbb{E}[f(X_T)] - \frac{1}{M}\sum_{i=1}^{M} f(\hat{X}_N^i)\right|,$$

which is a combination of the integration error and the approximation error.

$$\begin{aligned}&\left|\mathbb{E}[f(X_T)] - \frac{1}{M}\sum_{i=1}^{M} f(\hat{X}_N^{(i)})\right| \\ &\leq \underbrace{\left|\mathbb{E}[f(X_T)] - \mathbb{E}[f(\hat{X}_N)]\right|}_{=\text{ approximation error}} + \underbrace{\left|\mathbb{E}[f(\hat{X}_N)] - \frac{1}{M}\sum_{i=1}^{M} f(\hat{X}_N^{(i)})\right|}_{=\text{ integration error}}.\end{aligned}$$

For a given discretization and integration method, we can only further influence the errors by choosing the number of Monte Carlo paths M and the number of discretization steps N.
If we increase M, the integration error decreases, and if we increase N, the approximation error decreases. Normally one would choose M and N dependent of each other thus both have a similar influence on the total computational error.
In our case we just tried to increase N as far as possible to get an idea of the solution the hybrid scheme convergences to, since a high N should give us a good idea of the actual solution of the scheme.
Unfortunately, as we will see later, the hybrid scheme differs for $N = 16,384$ and $\kappa = 0, \ldots, 3$ only slightly in the 4^{th} decimal place. So we had to choose $M \approx 4$ million, to get to some degree a clear result in the 4^{th} decimal place, to be able to clearly distinguish the hybrid scheme for the different κ.
What we did computationally to receive those numbers of discretization steps and Monte Carlo paths is described under section 8.

5.4.1 Rate of convergence and experimental order of convergence

Our main goal is to determine the order of convergence γ for the weak error. It is known, that an empirical rate of convergence ($CN_i^{-\gamma}$) can be calculated as the slope of a regression line in a log-log plot [16]. If we calculate the empirical convergence rate, we get an approximation of the order of convergence. Here $(N_i)_{i\in I}$, $I = I_{\text{hybrid}}$, I_{exact} is our sequence of discretization steps given by $N_i = 2^i$ with $I_{\text{hybrid}} = \{1, \ldots, 14\}$ for the hybrid scheme and $I_{\text{exact}} = \{1, \ldots, 11\}$ for the exact scheme.

Note that the difference in the total number of discretizations steps is due to the computational time that is needed for the calculations.

We compute the regression parameters C and γ for the empirical convergence rate by a first degree polynomial fitting curve for the regression weak error$_i \sim CN_i^{-\gamma}$. Therefore we have taken the logarithm

$$\log(CN_i^{-\gamma}) = \log(C) - \gamma \log(N_i).$$

Then we use the *NumPy* function *polyfit* with $X = -\log((N_i)_{i \in I})$, $Y =$ log(weak error) and *degree = 1* to get the linear equation $Y = a + bX$ where $a = log(C)$ and $\gamma = b$. Hence we get an empirical solution for order of convergence γ.
In the following we will write ROC for the empirical rate of convergence.

A different approach is to use the experimental order of convergence (EOC). The EOC is a method from numerical mathematics to approximate the convergence rate of a given method. The EOC is e.g. described in [25, p. 137] and [40, 3.1.1]. Both use the EOC for approximating the weak error of a MC simulation.

Definition 11 (Experimental order of convergence (EOC)) Let n and $n-1$ be two successive discretization steps, $n = 1, \ldots, N$, and e_n^w, e_{n-1}^w the weak error for n or $n-1$ steps. The *experimental order of convergence (EOC)* is defined as

$$EOC(n, n-1) = \frac{\log(e_n^w(T)) - \log(e_{n-1}^w(T))}{log(n-1) - log(n)}.$$

The $EOC(n, n-1)$ is the slope of a line connecting the points (n, e_n^w) and $(n-1, e_{n-1}^w)$ in a figure with logarithmic scaled axes. Thus it is an experimental estimate of the order of convergence [25, p.137].

Remark 12 The EOC gives us an order of convergence from $n-1 \to n$, where $n-1$ and n are two successive discretization steps. Hence we don't get one approximation for the order of convergence for the whole simulation, but an approximation from one discretization step size to the next. If the scheme makes a jump, due for example to statistical errors, the EOC can also change drastic. On the other hand if the scheme convergences smoothly we would expect to have a relatively constant EOC.
On the other hand the ROC gives us one order of convergence for the whole scheme. Because we calculate the order of convergence as the slope of a linear regression line of the weak error, the ROC smooths the result. Hence individualized jumps won't have a big impact on the order of convergence.

5.4.2 Difference Monte Carlo Scheme

The idea of the difference Monte Carlo scheme is to directly compute an approximation of the weak error of a Monte Carlo scheme without having a reference solution. The difference Monte Carlo was for example used before by [20, p. 54] and [40, 3.2.4]. Instead of calculating the weak error of a given solution and a simulation result, i.e. calculating

$$|\mathbb{E}[f(X_T)] - \mathbb{E}[f(\hat{X}_T^N)]|$$

where $\mathbb{E}[f(X_T)]$ is the reference solution, N the maximum number of discretization steps and $\mathbb{E}[f(X_T^N)]$ the obtained solution by the Monte Carlo simulation, one takes two successive values of the discretization steps, i.e.

$$difMC_i = |\mathbb{E}[f(\hat{X}_T^{N_i})] - \mathbb{E}[f(\hat{X}_T^{N_{i-1}})]|,\ i = 1, \ldots, 14 \text{ respectivley } 12.$$

Remark 13 The difference Monte Carlo scheme will give us a slightly different weak error than the comparison with a reference solution $\mathbb{E}[f(X_T)]$, because

$$\begin{aligned} |\mathbb{E}[f(X_T)] - \mathbb{E}[f(\hat{X}_T^{N_i})]| &= |\mathbb{E}[f(X_T) - f(\hat{X}_T^{N_{i+1}})] + \mathbb{E}[f(\hat{X}_T^{N_{i+1}}) - f(\hat{X}_T^{N_i})]| \\ &\neq |\mathbb{E}[f(\hat{X}_T^{N_{i+1}})] - \mathbb{E}[f(\hat{X}_T^{N_i})]|, \end{aligned}$$

for given $i = 1, \ldots, N$.

6 Strong error of rough volatility models

Before we start with our analysis of the weak error we will present in this section an order of convergence for the strong error of rough volatility models.

In [33] A. Neuenkirch and T. Shalaiko proofed for a rough volatility model, which has the same volatility process (4.3.3) as the RSFV model, that the optimal rate of convergence for the strong error in respect to the root mean square error is N^{-H}, where N denotes the number of subintervals of the discretization. This optimal rate is obtained by the Euler method and an Euler-trapezoidal type scheme. The model they looked at has the form

$$S_t = S_0 e^{X_t} \tag{6.0.1}$$

$$X_t = -\frac{1}{2}\int_0^t \sigma_s^2 ds + \rho \int_0^t \sigma_s dZ_s, \tag{6.0.2}$$

where σ_s and Y_s are given by 4.3.2 respectively 4.3.3. Z_s is a standard Brownian motion given by

$$dZ_s = \rho dW_s + \sqrt{1-\rho^2} dW_s^{\perp},$$

where $\rho \in (-1, 1)$ is the correlation coefficient, W is a Brownian motion correlated with W^H, i.e.

$$\mathbb{E}[W_t^H W_s] = \rho^*(t,s),\ t \in \mathbb{R},\ s \geq 0$$

for some suitable, positive definite, function $\rho^* : \mathbb{R} \times [0,\infty) \to \mathbb{R}$. Further W^H and $W^{\perp}$ are independent, i.e.

$$\mathbb{E}\left[W_t^H W_s^{\perp}\right] = 0,\ t \in \mathbb{R}, s \geq 0.$$

In the following we will use the notation $\exp(Y_t)$ for σ_t, see 4.3.2, to clarify the dependence on Y_t. So X_t (6.0.2) can also be written as

$$X_t = -\frac{1}{2}\int_0^t e^{2Y_s} ds + \rho \int_0^t e^{Y_s} dWs + \sqrt{1-\rho^2}\int_0^t e^{Y_s} dW_s^{\perp},$$

which is the notation used in [33].

Besides the optimal convergence rate they proofed the existence of a lower bound for the convergence of the root mean square error. In the following we will give the main results from [33].

Remark 14 We just look at the mean square error (MSE) so we can leave out the square root in our notations. Therefore in the following theorems the rate of convergence is given by N^{-2H}. Taking the square root gives the mentioned error.

Let us first take a look at the lower bound of the MSE error, which is given by [33, Theorem 2].

The optimal mean square approximation of X_T using the discretization

$$W_0, W_{\frac{T}{N}}, \dots, W_T, W_0^{\perp}, W_{\frac{T}{N}}, \dots, W_T, Y_0, Y_{\frac{T}{N}}, \dots, Y_t$$

is given by

$$X_n^{\text{opt}} = \mathbb{E}\left[X_T | \mathcal{G}_N\right],$$

where $\mathcal{G}_N = \sigma(W_{\frac{kT}{N}}, W^{\perp}_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k = 0, \cdots N)$. With that we get the following lower bound for the strong error.

Theorem 1 *In the notation above the following holds*

$$\liminf_{N\to\infty} N^{2H}\mathbb{E}\left[\left(X_T - \mathbb{E}[X_T|\mathcal{G}_N]\right)^2\right] \geq (1-\rho^2)\frac{1}{(2H+1)(2H+2)}T^{2H+1}\theta^2\mathbb{E}[(e^{Y_0})^2].$$

If we look at the Euler method and Euler-trapezoidal scheme we receive the following [33, Theorem 1].

Theorem 2 *Suppose X_N^E and X_N^{Tr} be the Eluer method respectively the trapezoidal scheme, $N \geq 1$. Then it holds*

$$\begin{aligned}\mathbb{E}[X_N^E - X_T]^2 &= C_E * N^{-2H} + o(N^{-2H})\\ \mathbb{E}[X_N^{Tr} - X_T]^2 &= C_{Tr} * N^{-2H} + o(N^{-2H})\end{aligned}$$

with

$$\begin{aligned}C_E = C_E(\nu,\lambda,\theta,H,T) &= \frac{\theta^2\mathbb{E}[(e^{Y_0})^2]T^{2H+1}}{2H+1}\\ C_{rT} = C_{rT}(\nu,\lambda,\theta,H,T) &= \left(\frac{1}{2H+1} - \frac{1-\rho^2}{4}\right)T^{2H+1}\theta^2\mathbb{E}[e^{Y_0}]\\ &= C_E - (1-\rho^2)\frac{\theta^2\mathbb{E}[(e^{Y_0})^2]T^{2H+1}}{4}.\end{aligned}$$

Remark 15 For the trapezoidal scheme they used a slightly different version as we introduced in section 5.1, because they don't apply the trapezoidal rule to the integral $\rho \int_0^t e^{Y_s} dWs$. Hence they get

$$X_N^{Tr} = -\frac{1}{4}\sum_{k=0}^{N-1}\left(e^{2\Delta Y_k} + e^{2Y\Delta Y_{k+1}}\right)\Delta + \rho\sum_{k=0}^{N-1} e^{\Delta Y_k}\Delta W_k + \frac{1}{2}\sqrt{1-\rho^2}\sum_{k=0}^{N-1}\left(e^{\Delta Y_k} + e^{\Delta Y_{k+1}}\right)\Delta W_k^{\perp},$$

where $\Delta = \frac{T}{n}$ and $\Delta(\cdot)_k = (\cdot)_{\frac{(k+1)T}{n}} - (\cdot)_{\frac{kT}{n}}$, $k = 0, \ldots, n-1$.

Remark 16 The values for the error bounds can directly be obtained by the model parameters. In [33, Appendix, Theorem 1 and Lemma 3] it is shown, that

$$\theta^2 \mathbb{E}[e^{Y_0}]^2 = \theta^2 e^{2\nu} \exp\left(2\theta^2 \frac{\Gamma(2H+1)}{2\lambda^{2H}}\right).$$

Thus the values in theorem 1 and 2 can directly be calculated by the parameters of the volatility model Y (4.3.3).

6.1 Proof of the lower barrier for the strong error

In the following we will give the proof of theorem 1 [cp. 33].

Lemma 3 *Let $R = (R_t)_{t\in[0,T]}$ be a process, which is $(\mathcal{F}_t)_{t\in[0,T]}$ adapted, independent of $(W_t)_{t\in in[0,T]}$, and has root mean-square smoothness of order $\alpha \in (0,1)$, i.e. there exist $C > 0$, $\alpha \in (0,1)$, such that*

$$\left(\mathbb{E}\left[(R_t - R_s)^2\right]\right)^{\frac{1}{2}} \leq C|t-s|^{\alpha}, \ \ s,t \in [0,T].$$

Moreover, let $T_k = \frac{kT}{N}$, $k = 0, \ldots, N$ and

$$\begin{aligned}\bar{W}_t^N &= W_{t_k} + \left(\frac{Nt}{T} - k\right)\left(W_{t_{k+1}} - W_{t_k}\right) \\ &= W_{t_k} + \left(\frac{Nt}{T} - k\right)\Delta W_k, \ \ t \in [t_k, t_{k+1}], \ \ k = 0, \ldots, n-1.\end{aligned}$$

Then, it holds

$$\mathbb{E}\left[\left(\int_0^T R_s dW_s - \frac{1}{2}\sum_{k=0}^{N-1}(R_{t_k} + R_{t_{k+1}})\Delta W_k\right)^2\right] = \sum_{k=0}^{N-1}\int_{t_k}^{t_{k+1}} \mathbb{E}\left[\left(R_t - \frac{1}{2}(R_{t_k} + R_{r_{k+1}})ds\right)^2\right] dt$$

and

$$\mathbb{E}\left[\left(\int_0^T R_s dW_s - \int_0^T R_s d\bar{W}_s^n\right)^2\right] = \sum_{k=0}^{N-1}\int_{t_k}^{t_{k+1}} \mathbb{E}\left[\left(R_t - \frac{n}{T}\int_{t_k}^{t_{k+1}} R_s ds\right)^2\right] dt.$$

PROOF W.o.l.g. we will set $T = 1$. We only proof the second assertion, because we only need the second assertion for the proof of theorem 1.
Note that from the independence of W and R follows

$$\begin{aligned}\int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N &= \int_{t_k}^{t_{k+1}} R_s d(W_{s_k} + (Ns - k)\,\Delta W_k) \\ &= \int_{t_k}^{t_{k+1}} R_s dNs\Delta W_k = N\Delta W_k \int_{t_k}^{t_{k+1}} R_s ds.\end{aligned}$$

At first let us consider

$$\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s - \int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N\right)\left(\int_{t_l}^{t_{l+1}} R_s dW_s - \int_{t_l}^{t_{l+1}} R_s d\bar{W}_s^N\right)\right],$$

where $k \neq l$.
Due the independence of R and W and $\mathbb{E}\left[\Delta W_k \Delta W_l\right] = 0$ it holds

$$\begin{aligned}\mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N \int_{t_l}^{t_{l+1}} R_s d\bar{W}_s^N\right] &= \mathbb{E}\left[N\Delta W_k \int_{t_k}^{t_{k+1}} R_s ds\ N\Delta W_l \int_{t_l}^{t_{l+1}} R_s ds\right] \\ &= N^2 \mathbb{E}\left[\Delta W_k \Delta W_l\right] \mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s ds \int_{t_l}^{t_{l+1}} R_s ds\right] \\ &= 0.\end{aligned}$$

W.l.o.g be $k < l$. Then due to the adaptedness of R we get

$$\begin{aligned}\mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N \int_{t_l}^{t_{l+1}} R_s dW_s\right] &= \mathbb{E}\left[N\Delta W_k \int_{t_k}^{t_{k+1}} R_s ds \int_{t_l}^{t_{l+1}} R_s dW_s\right] \\ &= \mathbb{E}\left[N\Delta W_k \int_{t_k}^{t_{k+1}} R_s ds \mathbb{E}\left[\int_{t_l}^{t_{l+1}} R_s dW_s \middle| \mathcal{F}_{t_{k+1}}\right]\right] \\ &= 0.\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s dW_s \int_{t_l}^{t_{l+1}} R_s d\bar{W}_s^N\right] &= \mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s dW_s N\Delta W_l \int_{t_l}^{t_{l+1}} R_s ds\right] \\ &= \mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s dW_s \mathbb{E}\left[N\Delta W_l \int_{t_l}^{t_{l+1}} R_s ds \middle| \mathcal{F}_{t_{k+1}}\right]\right] \\ &= 0.\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s dW_s \int_{t_l}^{t_{l+1}} R_s dW_s\right] &= \mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s dW_s \mathbb{E}\left[\int_{t_l}^{t_{l+1}} R_s dW_s \middle| \mathcal{F}_{t_{k+1}}\right]\right] \\ &= 0.\end{aligned}$$

Hence we get for $k \neq l$

$$\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s - \int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N\right)\left(\int_{t_l}^{t_{l+1}} R_s dW_s - \int_{t_l}^{t_{l+1}} R_s d\bar{W}_s^N\right)\right] = 0.$$

Thus we further only need to consider the case where $k = l$. Thus we get

$$\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s - \int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N\right)^2\right] \tag{6.1.1}$$

$$= \mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s\right)^2\right] + \mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N\right)^2\right] - 2\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s \int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N\right)\right] \tag{6.1.2}$$

$$= \int_{t_k}^{t_{k+1}} \mathbb{E}\left[R_s^2\right] ds + N^2\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s ds\right)^2\right] - 2N\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s \int_{t_k}^{t_{k+1}} R_s ds \Delta W_k\right)\right]. \tag{6.1.3}$$

For the last term consider the approximation $S_M^{(k)} \xrightarrow[M\to\infty]{L^2} \int_{t_k}^{t_{k+1}} R_s dW_s$ with

$$S_M^{(k)} = \sum_{l=0}^{M-1} R_{s_l^M} \Delta W_{l^M},$$

where $(s_l^M)_{l=0}^M$ is a sequence of partitions of $[t_k, t_{k+1}]$ with grid size going to zero and $\Delta W_{l^M} = W_{s_{l+1}^M} - W_{s_l^M}$. The L^2 convergence holds due to the mean-square smoothness assumption. Then we have

$$\mathbb{E}\left[\int_{t_k}^{t_{k+1}} R_s dW_s \int_{t_k}^{t_{k+1}} R_s ds \Delta W_k\right] = \lim_{M\to\infty} \sum_{l=0} \mathbb{E}\left[R_{s_l^M} \int_{t_k}^{t_{k+1}} R_s ds\right] \mathbb{E}\left[\Delta W_{l^M} \Delta W_k\right]$$

$$= \lim_{M\to\infty} \sum_{l=0}^{M-1} \mathbb{E}\left[R_{S_l^M} \int_{t_k}^{t_{k+1}} R_s ds\right] (s_{l+1}^M - s_l^M) = \int_{t_k}^{t_{k+1}} \mathbb{E}\left[R_t \int_{t_k}^{t_{k+1}} R_s ds\right] dt.$$

If we use this in (6.1.3) then we get

$$\begin{aligned}
&\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s dW_s - \int_{t_k}^{t_{k+1}} R_s d\bar{W}_s^N\right)^2\right] \\
&= \int_{t_k}^{t_{k+1}} \mathbb{E}\left[R_s^2\right] ds + N^2\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} R_s ds\right)^2\right] - 2n \int_{t_k}^{t_{k+1}} \mathbb{E}\left[R_t \int_{t_k}^{t_{k+1}} R_s ds\right] \\
&= \int_{t_k}^{T_{k+1}} \mathbb{E}\left[R_t - N \int_{t_k}^{t_{k+1}} R_s ds\right].
\end{aligned}$$

Hence taking the sum over the subintervals yields the assertion. □

For the next lemma we need to introduce another σ-algebra

$$\mathcal{H}_N = \sigma(W_t, Y_t, t \geq 0, W^{\perp}_{\frac{kT}{N}}, k = 0, \ldots, N).$$

Further we use the notations

$$X_T^W = \rho \int_0^T e^{Y_s} dW_s$$

and

$$X_T^{W^{\perp}} = \sqrt{1-\rho^2} \int_0^T \sigma_s dW_s^{\perp}.$$

Lemma 4 *1. We have*

$$\mathbb{E}\left[X_T^{W^{\perp}} | \mathcal{G}_N\right] = \sqrt{1-\rho^2} \int_0^T \mathbb{E}\left[e^{Y_s} | W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k = 0, \ldots, N\right] d\overline{W^{\perp}}_s^N, \tag{6.1.4}$$

where

$$\overline{W^{\perp}}_t^N = W^{\perp}_{\frac{kT}{N}} + \left(\frac{Nt}{T} - k\right)(W^{\perp}_{\frac{(k+1)T}{N}} - W^{\perp}_{\frac{kT}{N}}),\ t \in \left[\frac{kT}{N}, \frac{(k+1)T}{N}\right],\ k = 0, \ldots, N-1,$$

and

$$\mathbb{E}\left[X_T^{W^{\perp}} | \mathcal{H}_N\right] = \sqrt{1-\rho^2} \int_0^T e^{Y_s} d\overline{W^{\perp}}_s^N.$$

2. It holds

$$\mathbb{E}\left[X_T^W | \mathcal{H}_N\right] = \rho W_t e^{Y_T} - \rho \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \mathbb{E}\left[\int_0^T W_s e^{Y_s^{\varepsilon}} (Y_s - Y_{s-\varepsilon}) ds | W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k = 0, \ldots, N\right]$$

with

$$Y_t^{\varepsilon} = \frac{1}{\varepsilon} \int_{t-\varepsilon}^t Y_s ds,\ t \in \mathbb{R},\ \varepsilon > 0.$$

Proof We start with the proof of 1.

1. Consider the L^2 approximation $S_M \xrightarrow[M \to \infty]{L^2} \frac{X_T^{W^{\perp}}}{\sqrt{1-\rho^2}}$, where

$$S_M = \sum_{l=0}^{M-1} e^{Y_{\frac{Tl}{M}}} (W^{\perp}_{\frac{(l+1)T}{M}} - W^{\perp}_{\frac{lT}{M}}),\ M \in \mathbb{N}.$$

Since for every sub-σ-algebra $\mathcal{G}$ of $\mathcal{F}$ and random variables X_n, X, $n \in \mathbb{N}$ it holds that [24, corollary 7.3.16]

$$\|\mathbb{E}[X|\mathcal{G}]\|_{L_2} \leq \|X\|_{L_2} \Rightarrow \|\mathbb{E}[X|\mathcal{G}] - \mathbb{E}[X_n|\mathcal{G}]\|_{L_2} \leq \|X - X_n\|_{L_2}$$

it follows

$$X_n \xrightarrow[N\to\infty]{L^2} X \Rightarrow \mathbb{E}\left[X_n|\mathcal{G}\right] \xrightarrow[N\to\infty]{L^2} \mathbb{E}[X|\mathcal{G}].$$

Therefore we have

$$\mathbb{E}[S_M|\mathcal{G}_N] \xrightarrow[M\to\infty]{L^2} \frac{1}{\sqrt{1-\rho^2}}\mathbb{E}\left[X_T^{W^\perp}|\mathcal{G}_N\right].$$

Obvious this is also true for $\mathcal{H}_N$. Hence we only need to show, that $\mathbb{E}[S_M|\mathcal{G}_N]$ convergences $\mathbb{P}$ - almost sure to the integral on the right hand side of (6.1.4). Then the proof is finished.

Because $W^\perp$ is independent of (W, Y) it holds

$$\begin{aligned}\mathbb{E}\left[S_M|\mathcal{G}_N\right] &= \sum_{l=0}^{M-1} \mathbb{E}\left[e^{\frac{Y_{Tl}}{M}}(W^\perp_{\frac{(l+1)T}{M}} - W^\perp_{\frac{lT}{M}})|W_{\frac{kT}{N}}, W^\perp_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k=0,\dots,n\right] \\ &= \mathbb{E}\left[e^{\frac{Y_{Tl}}{M}}|W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k=0,\dots,N\right]\mathbb{E}\left[(W^\perp_{\frac{(l+1)T}{M}} - W^\perp_{\frac{lT}{M}})|W^\perp_{\frac{kT}{N}}, k=0,\dots,N\right].\end{aligned}$$

Due to the normal correlation theorem, see for example [37, chapter 2], we have

$$\mathbb{E}\left[W^\perp_{\frac{(l+1)T}{M}} - W^\perp_{\frac{lT}{M}}|W^\perp_{\frac{kT}{N}}, k\dots,N\right] = \overline{W^\perp}^N_{\frac{(l+1)T}{M}} - \overline{W^\perp}^N_{\frac{lT}{M}}.$$

Therefore

$$\mathbb{E}\left[S_M|G_N\right] = \sum_{l=0}^{M-1} \mathbb{E}\left[e^{\frac{Y_{lT}}{M}}|W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k=0,\dots,N\right](\overline{W^\perp}^N_{\frac{(l+1)T}{M}} - \overline{W^\perp}^N_{\frac{lT}{M}}).$$

Since $\overline{W^\perp}^N_\cdot(\omega)$ is piecewise differentiable, it follows

$$\lim_{M\to\infty}\mathbb{E}[S_M|\mathcal{G}_N] = \int_0^T \mathbb{E}\left[e^{Y_s}|W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k=0,\dots,n\right] d\overline{W^\perp}^N_s.$$

In the same way follows, that

$$\lim_{M\to\infty}\mathbb{E}[S_M|\mathcal{H}_N] = \int_0^T e^{Y_s} d\overline{W^\perp}^N_s,$$

which finishes the proof.

Now we proof the 2. assertion. Therefore we introduce the following family of random variables

$$X_T^{W,\varepsilon} = \rho\int_0^T e^{Y_t^\varepsilon} dW_t,\ \varepsilon > 0.$$

Further note that

$$e^{Y_t^\varepsilon} - e^{Y_t} = \frac{e^{Y_t^\varepsilon} - e^{Y_t}}{Y_t^\varepsilon - Y_t}(Y_t^\varepsilon - Y_t) = \int_0^1 e^{\lambda Y_t^\varepsilon + (1-\lambda)Y_t} d\lambda (Y_t^\varepsilon - Y_t)$$

holds. Then together with the Itô isometry and Hölder's inequality we get

$$\begin{aligned}\mathbb{E}\left[\left(X_T^{W,\varepsilon} - X_T^W\right)^2\right] &= \rho^2 \int_0^T \mathbb{E}\left[\left(e^{Y_t^\varepsilon} - e^{Y_t}\right)^2\right] = \rho^2 \int_0^T \mathbb{E}\left[\left(\int_0^1 e^{\lambda Y_t^\varepsilon + (1-\lambda)Y_t} d\lambda (Y_t^\varepsilon - Y_t)\right)^2\right] \\ &\leq \int_0^T \mathbb{E}\left[\left(\int_0^1 e^{\lambda Y_t^\varepsilon + (1-\lambda)Y_t} d\lambda\right)^4\right]^{\frac{1}{2}} \mathbb{E}\left[(Y_t^\varepsilon - Y_t)^4\right]^{\frac{1}{2}} dt.\end{aligned}$$

Note that $\rho \in (-1, 1)$ and thus $\rho^2 \leq 1$. Since

$$|Y_t^\varepsilon| \leq \sup_{t\in[-1,T]} |Y_t|,\ t \in [0, T], \varepsilon(0, 1],$$

it follows that

$$\left(\int_0^1 e^{\lambda Y_t^\varepsilon + (1-\lambda)Y_t} d\lambda\right)^4 \leq \left(\int_0^1 e^{\lambda \sup_{t\in[-1,T]} |Y_t| + (1-\lambda) \sup_{t\in[-1,T]} |Y_t|} d\lambda\right)^4 = e^{4 \sup_{t\in[-1,T]} |Y_t|}.$$

Therefore we have

$$\mathbb{E}\left[(X_T^{W,\varepsilon} - X_T^W)^2\right] \leq C \int_0^T \mathbb{E}\left[(Y_t^\varepsilon - Y_t)^4\right]^{\frac{1}{2}}$$

with

$$C^2 = \mathbb{E}\left[e^{4 \sup_{t\in[-1,T]} |Y_t|}\right] < \infty$$

due to Fernique's theorem [27, theorem 4.1] . Finally we obtain

$$\mathbb{E}\left[(Y_t^\varepsilon - Y_t)^4\right]^{\frac{1}{2}} = c\mathbb{E}\left[Y_t^\varepsilon - Y_t\right]^2 \leq c \sup_{s\in[t-\varepsilon,t]} \mathbb{E}\left[Y_t - Y_s\right]^2,\ t \in [0, T],$$

for some constant $c > 0$ and so lemma 2 implies

$$\lim_{\varepsilon\to 0} \int_0^T \mathbb{E}\left[(Y_t^\varepsilon - Y_t)^4\right]^{\frac{1}{2}} = 0.$$

Hence it follows that

$$\lim_{\varepsilon\to 0} \mathbb{E}\left[(X_T^{W,\varepsilon} - X_T^W)^2\right] \leq \lim_{\varepsilon\to 0} C \int_0^T \mathbb{E}\left[(Y_t^\varepsilon - Y_t)^4\right]^{\frac{1}{2}} = 0.$$

Thus we get

$$\mathbb{E}[X_T^W | \mathcal{G}_N] = \lim_{\varepsilon\to 0} \mathbb{E}[X_T^{W,\varepsilon} | \mathcal{G}_N]. \tag{6.1.5}$$

With the formula for the integration by parts and the fact that for all $\varepsilon > 0$ the map $[0,T] \ni t \mapsto Y_t^\varepsilon(\omega) \in \mathbb{R}$ is differentiable we get the following formula for $X_T^{W,\varepsilon}$

$$X_T^{W,\varepsilon} = \rho \int_0^T e^{Y_t^\varepsilon} dW_s = \rho e^{Y_T^\varepsilon} W_T + \rho e^{Y_0^\varepsilon} W_0 - \rho \int_0^T W_t de^{Y_t^\varepsilon} \tag{6.1.6}$$

$$= \rho e^{Y_T} W_t - \frac{\rho}{\varepsilon} \int_0^T W_t (Y_t - Y_{t-\varepsilon}) e^{Y_t^\varepsilon} dt. \tag{6.1.7}$$

Thus if we combine (6.1.5) and (6.1.7) and use the independence of (W,Y) and $W^\perp$ we get

$$\begin{aligned}\mathbb{E}[X_T^W|\mathcal{G}_N] &= \lim_{\varepsilon\to 0} \mathbb{E}[X_T^{W,\varepsilon}|\mathcal{G}_N] \\ &= \rho W_T e^{Y_T} - \rho \lim_{\varepsilon\to 0} \mathbb{E}\left[\int_0^T W_t(Y_t - Y_{t-\varepsilon}) e^{Y_t^\varepsilon} dt \Big| W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k = 0,\dots,N\right],\end{aligned}$$

which finishes the proof of the 2. assertion and therefore the proof of the lemma. □

If we use the definition of $\overline{W^\perp}_s^N$, we can rewrite (6.1.4) to

$$\mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N] = \sqrt{1-\rho^2} \sum_{l=0}^{N-1} \frac{N}{T} \Delta W^\perp{}_l \int_{\frac{lT}{N}}^{\frac{(l+1)T}{N}} \mathbb{E}\left[e^{Y_s}|W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k = 0,\dots,N\right] ds.$$

Further we get $\mathbb{E}[X_T^W|\mathcal{G}_N] = \mathbb{E}\left[X_T^W|W_{\frac{kT}{N}}, Y_{\frac{kT}{N}}, k = 0,\dots,N\right]$. Therefore we obtain again with the independence of $W^\perp$ from (W,Y)

- $\mathbb{E}\left[\mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N] X_T^W\right] = \mathbb{E}\left[X_T^W\right] \mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N] = 0$
- $\mathbb{E}\left[\mathbb{E}[X_T^W|\mathcal{G}_N] X_T^{W^\perp}\right] = \mathbb{E}\left[X_T^{W^\perp}\right] \mathbb{E}[X_T^W|\mathcal{G}_N] = 0$
- $\mathbb{E}\left[\mathbb{E}[X_T^W|\mathcal{G}_N] \mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N]\right] = \mathbb{E}\left[\mathbb{E}[X_T^W|\mathcal{G}_N]\right] \mathbb{E}\left[\mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N]\right] = 0$
- $\mathbb{E}[X_T^{W^\perp} X_T^W] = \mathbb{E}[X_T^{W^\perp}] \mathbb{E}[X_T^W] = 0.$

Thus it holds

$$\mathbb{E}\left[\left(X_T^{W^\perp} - \mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N]\right)\left(X_T^W - \mathbb{E}[X_T^W|\mathcal{G}_N]\right)\right] = 0.$$

Hence we have

$$\mathbb{E}\left[\left(X_T^{W^\perp} + X_T^W - \mathbb{E}[X_T^{W^\perp} + X_T^W|\mathcal{G}_N]\right)^2\right] \tag{6.1.8}$$

$$= \mathbb{E}\left[(X_T^{W^\perp} - \mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N])^2\right] + \mathbb{E}\left[(X_T^W - \mathbb{E}[X_T^W|\mathcal{G}_N])^2\right] \tag{6.1.9}$$

$$\geq \mathbb{E}\left[(X_T^{W^\perp}) - \mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N]\right]. \tag{6.1.10}$$

Before we can finally give the proof of theorem 1 we need to introduce the process $Z_t^{(a)}$, which is defined as follows

$$Z_t^{(a)} := e^{a(Y_t-\nu)},\ t \in \mathbb{R}.$$

For the calculation of the expected value we will use the moment generating function $M_X(t) = \mathbb{E}[e^{tX}]$ [24, chapter 6] of normal distributed random variables, because for a normal random variable $X \sim \mathcal{N}(\nu, \sigma^2)$ we know that

$$\mathbb{E}[e^{tX}] = \exp(\nu t + \frac{t^2\sigma^2}{2})$$

holds [24, chapter 8]. Because $a(Y_t - \nu) \sim \mathcal{N}(0, a^2 Var(Y_t))$ we therefore get

$$\mathbb{E}[Z_t^{(a)}] = \mathbb{E}[e^{a(Y_t-\nu)}] = \exp\left(\frac{a^2}{2}Var(Y_t)\right).$$

Later we will also need $\mathbb{E}[(Z_t^{(a)})^2]$. With the same method and the fact that $2a(Y_t - \nu) \sim \mathcal{N}(0, 4a^2 Var(Y_t))$ holds we get

$$\mathbb{E}[(Z_t^{(a)})^2] = \mathbb{E}[e^{2a(Y_t-\nu)}] = \exp\left(2a^2 Var(Y_t)\right).$$

Further the covariance of $Z_t^{(a)}$ and $Z_s^{(a)}$ is given by

$$\begin{aligned}
Cov(Z_t^{(a)}, Z_s^{(a)}) &= \mathbb{E}[Z_t^{(a)} Z_s^{(a)}] - \mathbb{E}[Z_t^{(a)}]\mathbb{E}[Z_s^{(a)}] \\
&= \mathbb{E}[\exp(a(Y_t - \nu))\exp(a(Y_s - \nu))] - \exp\left(\frac{a^2}{2}Var(Y_t)\right)\exp\left(\frac{a^2}{2}Var(Y_s)\right) \\
&= \mathbb{E}[\exp(a(Y_t + Y_s - 2\nu))] - \exp(a^2 Var(Y_t)) \\
&= \exp\left(\frac{a^2}{2}Var(Y_t + Y_s)\right) - \exp(a^2 Var(Y_t)) \\
&= \exp\left(\frac{a^2}{2}(Var(Y_t) + Var(Y_s) + 2Cov(Y_t, Y_s)\right) - \exp(a^2 Var(Y_t)) \\
&= \exp(a^2(Var(Y_t) + Cov(Y_t, Y_s)) - \exp(a^2 Var(Y_t)) \\
&= \exp(a^2 Var(Y_t))(\exp(a^2 Cov(Y_t, Y_s)) - 1)
\end{aligned}$$

due to $a(Y_t + Y_s - 2\nu) \sim \mathcal{N}(0, a^2 Var(Y_t + Y_s))$ and the formula for the variance of a sum [24, lemma 3.5.2]. We can also write the covariance in terms of (4.3.4) [33, p. 7]

$$Cov(Z_t^a, Z_s^a) = R_{Z^{(a)}}(|t - s|).$$

The process $Z_t^{(a)}$ fulfils the following two lemmas [33,Lemma 3 & 4].

Lemma 5 (33, Lemma 3) *Let $a \neq 0$. We have $R_{Z^{(a)}} \in C^\infty((0,\infty);\mathbb{R})$ and*

$$R_{Z^{(a)}}(\tau) = c_0 - c_1\tau^{2H} + o(\tau^{2H}),\ \tau \to 0,$$

with

$$c_0 = \exp(a^2 Var(Y_t))\left(\exp(a^2 Var(Y_t)) - 1\right)$$

and

$$c_1 = \frac{a^2\theta^2}{2}\exp\left(2a^2 Var(Y_t)\right) = \frac{a^2\theta^2}{2}\mathbb{E}\left[(Z_t^{(a)})^2\right].$$

Lemma 6 (33, Lemma 4) *Let $a \neq 0$, $s \leq u \leq t$ and be c_1 as in lemma 5. Then we have*

$$\mathbb{E}\left[(Z_t^{(a)} - Z_s^{(a)})^2\right] = 2c_1|t-s|^{2H} + o(|t-s|^{2H}),\ |t-s| \to 0$$

and

$$\mathbb{E}\left[\left(Z_u^{(a)} - \frac{1}{2}(Z_s^{(a)} + Z_t^{(a)})\right)^2\right] = c_1(|t-u|^{2H} + |s-u|^{2H} - \frac{1}{2}|t-s|^{2H}) + o(|t-s|^{2H}),\ |t-s| \to 0.$$

Further it holds

$$\mathbb{E}\left[(Z_t^{(a)} - Z_u^{(a)})(Z_t^{(a)} - Z_s^{(a)})\right] = c_1(|t-s|^{2H} + |t-u|^{2H} - |u-s|^{2H}) + o(|t-s|^{2H}),\ |t-s| \to 0.$$

Now we can finally give the proof of theorem 1.

Proof First let us the following notation [33, p. 8]

$$X_T^{RS} = \int_0^T e^{2Y_s}ds = e^{2\nu} = \int_0^T Z_s^{(2)}ds.$$

Since $Z^{(a)}$ is stationary and the covariance function is infinitely differentiable from zero and admits the expansion given in lemma 5, we can apply a result form Benhenni [7, theorem 1] to get

$$\mathbb{E}\left[\left(X_T^{RS} - \frac{1}{2}\sum_{k=0}^{N-1}\left(e^{2\Delta Y_k} + e^{2\Delta Y_{k+1}}\right)\Delta\right)^2\right] = O(\Delta^{1+2H}). \tag{6.1.11}$$

Since

$$\frac{1}{2}\sum_{k=0}^{N-1}\left(e^{2\Delta Y_k} + e^{2\Delta Y_{(k+1)}}\right)\Delta = \sum_{k=0}^{N-1} e^{2\Delta Y_k}\Delta + \frac{1}{2}\left(e^{2Y_T} - e^{2Y_0}\right)\Delta$$

and $H < \frac{1}{2}$ it also holds

$$\mathbb{E}\left[\left(X_T^{RS} - \sum_{k=0}^{N-1} e^{2\Delta Y_k}\Delta\right)^2\right] = O(\Delta^{1+2H}).$$

Therefore for the Riemann integral part of X_T we have

$$\lim_{N\to\infty} N^{2H}\mathbb{E}\left[\left(X_T^{RS} - \mathbb{E}\left[X_T^{RS}|\mathcal{G}_N\right]\right)^2\right] = 0.$$

Thereby and with $\mathcal{G}_N \subset \mathcal{H}_n$ and the inequality (6.1.10) it holds

$$\begin{aligned}\liminf_{N\to\infty} N^{2H}\mathbb{E}\left[(X_T - \mathbb{E}[X_T|\mathcal{G}_N])^2\right] &\geq \liminf_{N\to\infty} N^{2H}\mathbb{E}\left[(X_T^{W^\perp} - \mathbb{E}[X_T^{W^\perp}|\mathcal{G}_N])^2\right] \\ &\geq \liminf_{N\to\infty} N^{2H}\mathbb{E}\left[(X_T^{W^\perp} - \mathbb{E}[X_T^{W^\perp}|\mathcal{H}_N])^2\right].\end{aligned}$$

Then with the lemmata 4 and 3 it follows that

$$\mathbb{E}\left[(X_T^{W^\perp} - \mathbb{E}[X_T^{W^\perp}|\mathcal{H}_N])^2\right] = (1-\rho^2)\frac{N^2}{T^2}\sum_{k=0}^{N-1}\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \mathbb{E}\left[\left(\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} (e^{Y_t} - e^{Y_s})ds\right)^2\right] dt.$$

If we use the process $Z_t^{(1)}$ to rewrite $(e^{Y_t} - e^{Y_s})$ we can apply lemma 6 and obtain

$$\begin{aligned}&\mathbb{E}\left[\left(\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} (e^{Y_t} - e^{Y_s})ds\right)^2\right] \\ &= e^{2\nu}\int_{\frac{kT}{N}}^{\frac{(k+1)T)}{N}}\int_{\frac{kT}{N}}^{\frac{(k+1)T)}{N}} \mathbb{E}\left[(Z_t^{(1)} - Z_{s_1}^{(1)})(Z_t^{(1)} - Z_{s_2}^{(1)})\right] ds_1 ds_2 \\ &= e^{2\nu}c_1\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}}\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \left(|t-s_1|^{2H} + |t-s_2|^{2H} + |t-s_2|^{2H} - |s_1-s_2|^{2H}\right) ds_1 ds_2 + o(N^{-2H-2}).\end{aligned}$$

Thus it follows

$$\begin{aligned}&\frac{1}{c_1 e^{2\nu}}\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \mathbb{E}\left[\left(\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} (e^{Y_t} - e^{Y_s})ds\right)^2\right] dt \\ &= \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}}\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}}\int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} (|t-s_1|^{2H} + |t-s_2|^{2H} - |s_1-s_2|^{2H}) ds_1 ds_2 dt + o(n^{-2H-3}) \\ &= \int_0^{\frac{T}{N}}\int_0^{\frac{T}{N}}\int_0^{\frac{T}{N}} |s_1-s_2|^{2H} ds_1 ds_2 dt + o(N^{-2H-3}) \\ &= \frac{2}{(2H+1)(2H+2)}\left(\frac{T}{N}\right)^{2H+3} + o(N^{-2H-3}).\end{aligned}$$

Summing up and using $c_1 = \frac{\theta^2}{2}\mathbb{E}\left[(Z_0^{(1)})^2\right] = \frac{\theta^2}{2}e^{-2\nu}\mathbb{E}[(e^{Y_0})^2]$ yields the assertion. □

6.2 Proof of the convergence rate of the Euler and trapezoidal scheme

In the previous section we have introduced all lemmas we need to give the proof for theorem 1.

PROOF (33, PP. 8-9) The first Riemann integral part of X_T was already considered at the beginning of the proof of theorem 2.
Therefore we further only consider the parts X_T^W and $X_T^{W^\perp}$. Using $Y_t = e^\nu Z_t^{(1)}$, the Itô isometry and lemma 6 we get

$$\begin{aligned}
&\mathbb{E}\left[\left(\int_0^T e^{Y_s} dW_s - \sum_{k=0}^{N-1} e^{Y_{\frac{kT}{N}}} \Delta W_k\right)^2\right] = \mathbb{E}\left[\left(\sum_{k=0}^{N-1} \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} (e^{Y_s} - e^{Y_{\frac{kT}{N}}}) dW_s\right)^2\right] \\
&= e^{2\nu} \sum_{k=0}^{N-1} \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \mathbb{E}\left[(Z_s^{(1)} - Z_{\frac{kT}{N}})\right] \\
&= 2c_1 e^{2\nu} \sum_{k=0}^{N-1} \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \left((s - \tfrac{kT}{N})^{2H} + o(|s - \tfrac{kT}{N}|^{2H})\right) ds \\
&= \frac{2c_1 e^{2\nu}}{2H+1} N \left(\frac{T}{N}\right)^{2H+1} (1 + o(1)).
\end{aligned}$$

Analogously

$$\mathbb{E}\left[\left(\int_0^T e^{Y_s} dW_s^\perp - \sum_{k=0}^{N-1} e^{Y_{\frac{kT}{N}}} \Delta W_k^\perp\right)^2\right] = \frac{2c_1 e^{2\nu}}{2H+1} N \left(\frac{T}{N}\right)^{2H+1} (1 + o(1)).$$

Summing up both terms and using the orthogonality of the stochastic integrals we end up with the statement for the Euler scheme.

For the analysis of the trapezoidal scheme we have by lemma 3

$$\begin{aligned}
&\mathbb{E}\left[\left(\int_0^T Z_s^{(1)} dW_s^\perp - \frac{1}{2} \sum_{k=0}^{N-1} (Z_{\frac{kT}{N}}^{(1)} + Z_{\frac{(k+1)T}{N}}^{(1)}) \Delta W_k\right)^2\right] \\
&= \sum_{k=0}^{N-1} \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \mathbb{E}\left[\left(Z_t^{(1)} - \frac{1}{2}(Z_{\frac{kT}{N}}^{(1)} + Z_{\frac{(k+1)T}{N}}^{(1)})\right)^2\right] dt.
\end{aligned}$$

Due to lemma 6 we have

$$\sum_{k=0}^{N-1} \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \mathbb{E}\left[\left(Z_t^{(1)} - \frac{1}{2}(Z_{\frac{kT}{N}}^{(1)} + Z_{\frac{(k+1)T}{N}}^{(1)})\right)^2\right] dt$$
$$= c_1 \sum_{k=0}^{N-1} \int_{\frac{kT}{N}}^{\frac{(k+1)T}{N}} \left(|t - \tfrac{kT}{N}|^{2H} + |\tfrac{(k+1)T}{N} - t|^{2H} - \frac{1}{2}\left(\tfrac{T}{N}\right)^{2H} + o(\left(\tfrac{T}{N}\right)^{2H})\right) dt$$
$$= c_1 \left(\frac{2}{2H+1} - \frac{1}{2}\right) N \left(\frac{T}{N}\right)^{2H} + o(\left(\tfrac{T}{N}\right)^{2H}).$$

Using this estimate, taking into account the correlation and the Euler estimate for X_T^W, we obtain our assertion. $\square$.

7 Simulation of the rBergomi model

In the description of the simulation we follow [4, section 4]. To simplify the notation the simulation starts at $t = 0$ and the reference to the pricing measure $\mathbb{Q}$ is dropped. Then the model becomes

$$S_t = S_0 \exp\left(\int_0^t \sqrt{v_u} dZ_u\right) \tag{7.0.1}$$

$$v_u = \xi_0(u)\mathcal{E}\left(\eta\sqrt{2H}\int_0^u \frac{1}{(u-s)^\gamma} dW_s\right) = \xi_0(u)\mathcal{E}\left(\eta\tilde{W}_u\right), \tag{7.0.2}$$

where $\tilde{W}$ is a truncated Brownian semi-stationary process and

$$dZ_t = \rho dW_t + \sqrt{1-\rho^2} dW_t^\perp. \tag{7.0.3}$$

7.1 Exact simulation

An exact method for the simulation of the rBergomi model [4, section 4] is to construct the joint covariance matrix of $\tilde{W}$ and Z from 7.0.2 respectively 7.0.3, then calculate its Cholesky decomposition and thereby random variables $(\tilde{W}, Z)$ with the correct joint distribution. The joint covariance of $(\tilde{W}, Z)$ is given by

$$\begin{cases} \mathbb{V}ar(\tilde{W}_t) = t^{2H} & , \ t \geq 0 \\ Cov(\tilde{W}_t, \tilde{W}_s) = t^{2H} G(\frac{s}{t}) & , \ s > t \geq 0 \\ Cov(\tilde{W}_t, Z_s) = \rho D_H (t^{H+\frac{1}{2}} - (t - \min(t,s)^{H+\frac{1}{2}})) & , \ t, s \geq 0 \\ Cov(Z_t, Z_s) = \min(t,s) & , \ t, s \geq 0 \end{cases} \tag{7.1.1}$$

with

$$D_H = \frac{\sqrt{2H}}{H + \frac{1}{2}}$$

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

$$G(x) = 2H \int_0^1 \frac{1}{(1-s)^\gamma (x-s)^\gamma} ds. \tag{7.1.2}$$

In the literature exist two different solutions for the integral 7.1.2:

1. a version by Bayer et al [4, p. 15]

$$G_1(x) = \frac{1-2\gamma}{1-\gamma} x^\gamma {}_2F_1(1, \gamma, 2-\gamma, x) \tag{7.1.3}$$

2. and a second version by Bayer et al [5, p. 9]

$$G_2(x) = 2H\left(\frac{1}{1-\gamma}x^{-\gamma} + \frac{\gamma}{1-\gamma}x^{-(1+\gamma)}\frac{1}{2-\gamma}{}_2F_1\left(1, 1+\gamma, 3-\gamma, x^{-1}\right)\right). \tag{7.1.4}$$

Furthermore we calculated a third version:

$$G_3(x) = 2H\int_0^1 \frac{1}{(1-s)^\gamma(x-s)^\gamma}ds = \begin{cases} 1, & x = 1 \\ \frac{1-2\gamma}{1-\gamma}x^{-\gamma}{}_2F_1(\gamma, 1, 2-\gamma, \frac{1}{x}), & x > 1. \end{cases} \tag{7.1.5}$$

In the following we will give a short proof of the equation 7.1.5. For the proof we will use Lemma 4.3 from [8].

Lemma 7 *For all $\alpha \in (-1, \infty)$ and $0 \leq a < b$,*

$$\int_0^a (a-x)^\alpha(b-x)^\alpha dx = \frac{a^{\alpha+1}b^\alpha}{\alpha+1}{}_2F_1\left(-\alpha, 1, \alpha+2, \frac{a}{b}\right).$$

Proof If $a = 0$ it's clear that the equation holds. Hence w.l.o.g. be $a > 0$. By substituting $y = \frac{x}{a}$ we receive

$$\begin{aligned}\int_0^a (a-x)^\alpha(b-x)^\alpha &= a^\alpha b^\alpha \int_0^a \left(\frac{a-x}{a}\right)^\alpha \left(\frac{b-x}{b}\right)^\alpha dx \\ &= a^\alpha b^\alpha \int_0^a \left(1-\frac{x}{a}\right)^\alpha \left(1-\frac{x}{b}\right)^\alpha dx \\ &= a^{\alpha+1}b^\alpha \int_0^1 (1-y)^\alpha \left(1-\frac{a}{b}y\right)^\alpha dy.\end{aligned}$$

Using Euler's formula, which is defined for $|z| < 1$ and for the real parts of b and c holds $c > b > 0$,

$${}_2F_1(a, b; c; z) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)}\int_0^1 t^{b-1}(1-t)^{c-b-1}(1-tz)^{-a}dt$$

form [14, Section 2.1.3, Eq. (10)] with $a = -\alpha,\ b = 1,\ c = \alpha+2$ and $z = \frac{a}{b}$ gives us

$$\int_0^1 (1-y)^\alpha \left(1-\frac{a}{b}y\right)^\alpha dy = \frac{\Gamma(1)\Gamma(\alpha+1)}{\Gamma(\alpha+2)}{}_2F_1\left(-\alpha, 1, \alpha+2, \frac{a}{b}\right).$$

Note that this is valid since $\alpha + 2 > 1 > 0$ and $0 < \frac{a}{b} < 1$ under our assumption.
By using the beta function B given for $x, y > 0$ by

$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt$$

[14, Sect. 1.5, Eq. (1)] and the equation [14, Sect. 1.5, Eq. (5)]

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

we get

$$\frac{\Gamma(1)\Gamma(\alpha+1)}{\Gamma(\alpha+2)} = B(1,\alpha+1) = \int_0^1 (1-x)^\alpha dx = \frac{1}{\alpha+1}.$$

Hence combining those we get

$$\int_0^a (a-x)^\alpha (b-x)^\alpha dx = \frac{a^{\alpha+1}b^\alpha}{\alpha+1} {}_2F_1\left(-\alpha, 1, \alpha+2, \frac{a}{b}\right).$$

□

Now we can proof equation 7.1.5.

Proof Be $x \geq 1$ and $\gamma = \frac{1}{2} - H$. First it holds that

$$\gamma = \frac{1}{2} - H \Leftrightarrow H = \frac{1}{2} - \gamma.$$

Furthermore we can write $G(x)$ in the following form

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

Therefore for $x = 1$ we get

$$\begin{aligned} G(1) &= 2H \int_0^1 (1-s)^{-\gamma}(1-s)^{-\gamma} ds = 2H \int_0^1 (1-s)^{-2\gamma} \\ &= 2H \frac{1}{1-2\gamma} = \frac{2\left(\frac{1}{2}-\gamma\right)}{1-2\gamma} = 1. \end{aligned}$$

So further we only need to consider the case $x > 1$. Note that since $H \in (0, \frac{1}{2})$ it holds that $\gamma \in (0, \frac{1}{2})$ and $-\gamma \in (-\frac{1}{2}, 0)$. With $\alpha = -\gamma$, $a = 1$ and $b = x$

we can use Lemma 7 for the integral since $-\gamma \in (-1, \infty)$ and $0 \leq 1 < x$. Therefore all requirements are fulfilled. With Lemma 7 we get

$$\int_0^1 (1-s)^{-\gamma}(x-s)^{-\gamma} ds = \frac{x^{-\gamma}}{1-\gamma} {}_2F_1\left(\gamma, 1, 2-\gamma, \frac{1}{x}\right)$$

and so

$$G(x) = 2H\frac{x^{-\gamma}}{1-\gamma} {}_2F_1\left(\gamma, 1, 2-\gamma, \frac{1}{x}\right) = \frac{1-2\gamma}{1-\gamma} x^{-\gamma} {}_2F_1\left(\gamma, 1, 2-\gamma, \frac{1}{x}\right).$$

□

Our new version of $G(x)$ allows us to combine the first two lines of the covariance matrix 7.1.1 to

$$Cov(\tilde{W}_t, \tilde{W}_s) = t^{2H} G_3\left(\frac{s}{t}\right), \ s \geq t \geq 0,$$

since

$$Var(W_s) = Cov(W_s W_s) = s^{2H} G_3\left(\frac{s}{s}\right) = s^{2H} G_3(1) = s^{2H}.$$

For a comparison of the three different solutions for $G(x)$ we wrote a short Python file, given below, to see if the three functions for $G(x)$ are actually different or just reformulations of one another. Table 1 contains the test results for $G_1(x)$, $G_2(x)$ and $G_3(x)$.

x	1.0	1.5	2	2.5	3
$G_1(x)$	1.0	0.3246 - 0.3053i	0.2701 - 0.3231i	0.2407 - 0.3315i	0.2214 - 0.3364j
$G_2(x)$	1.0	0.271361	0.218081	0.189401	0.170504
$G_3(x)$	1.0	0.271361	0.218081	0.189401	0.170504

Table 1: Comparison of G_1, G_2 and G_3

We see that the values for G_2 and G_3 are identical, while G_1 returns complex numbers. Hence we conclude that the formula for G_1 gives us wrong results, because the integral 7.1.2 should give us real numbers. Since the formulas for G_1 and G_3 look very a like, it can be that the formula for G_1 just contains a typo in the article, because the only differences between G_1 and G_3 are that in G_1 we have x^{γ} and in ${}_2F_1(a, b; c; z)$ $z = x$ and in G_2 we have $x^{-\gamma}$ and $z = \frac{1}{x}$.
Note that the complex numbers in G_1 came from the ${}_2F_1$ function, which we calculated with the mpmath package for Python [21]. For table 1 we used the following Python code.

```
import mpmath

def G1(x,gamma):
        g = ((1.-2.*gamma)/(1. - gamma))*(x**gamma)*
        mpmath.hyp2f1(1.,gamma, 2.-gamma,x)
        return g


def G2(x,H,gamma):
        g = 2*H*((1./(1.-gamma))*x**(-1.*gamma) +
            (gamma/(1-gamma))*x**(-1.*(1.+gamma))*(1./(2.-gamma)
                )*mpmath.hyp2f1(1.,1.+gamma,3.-gamma,x**(-1.)))
        return g

def G3(x,gamma):
        if x == 1:
                g = 1
        else:
                g = 2*H*(1./(1.-gamma))*(x**(-1.*gamma))*
                    mpmath.hyp2f1(gamma,1.,2.-gamma,x**(-1.))
        return g

if __name__ == '__main__':
        H = 0.07
        gamma = 0.43
        test = [1.,1.5,2.,2.5,3.]
        for x in test:
                print('x = ', x)
                print('G_1(x) = ', G1(x,gamma))
                print('G_2(x) = ', G2(x,H,gamma))
                print('G_3(x) = ', G3(x,gamma))
```

Remark 17 We further checked the numbers of the ${}_2F_1$ function with wolfram alpha. We received the same results. Furthermore if we use the scipy.special package for Python, we get the same results for G_2 and G_3, while the package isn't able to calculate values for G_1 if $x > 1$. This is highly due to the fact, that ${}_2F_1(a, b; c; z)$ only convergences absolutely for $|z| < 1$ [14, p. 57], while for $|z| > 1$ it has in general a continuation to the complex plane [26, section 2]. Therefore for $x > 1$, if we use G_1, we enter the complex plane, because $z = x \geq 1$. On the other hand in G_2 and G_3 $z = \frac{1}{x} \leq 1$. This guarantees the absolute convergence of ${}_2F_1$ and that we get real numbers as results.

Because the matrix doesn't have any 0 values, if $\rho \neq 0$, the computational effort for the exact simulation is quite high and the algorithm is very slow.

A different approach for the simulation of the rBergomi model, is to use the so-called hybrid scheme. This model needs less computational effort. Therefore we will describe the hybrid scheme in the next subsection.

7.2 Hybrid scheme

The hybrid scheme was developed by M. Bennedsen et al. [8] for the simulation of BSS 4.4.4 respectably TBSS 4.4.5 processes with kernel functions, that are similar to a power function near zero, i.e. $g(x)$ is similar to x^α for some $\alpha \in (-\frac{1}{2}, \frac{1}{2})$ when $x > 0$ is near zero. Because $\tilde{W}$ in the rBergomi model is a TBSS process and the kernel function satisfies the necessary conditions, we can simulate $\tilde{W}$ by using the hybrid scheme [8, 2.5 and 3.3].
In the following we give a recap of the hybrid scheme for TBSS processes from [8, 2.5]. In general the hybrid scheme is defined for BSS processes, but for the simulation of the rBergomi model we only need the hybrid Scheme for TBSS processes.

For the description of the hybrid Scheme we first need the definition of slowly varying functions [8, Section 2.1].

Definition 12 (Slowly varying at 0) A measurable function $L : (0, 1] \to [0, \infty)$ is *slowly varying at 0* if for any $t > 0$ holds

$$\lim_{x \to 0} \frac{L(tx)}{L(x)} = 1.$$

Now we can describe the requirements for the hybrid scheme.
Assume we have a TBSS process X_t as defined in 4.4.5

$$X_t = \int_0^t g(t-s)\sigma_s dWs,$$

where the kernel function $g : (0, \infty) \to [0, \infty)$ satisfies the following conditions [8, p. 935]:

1. For some $\alpha \in (-\frac{1}{2}, \frac{1}{2})\backslash\{0\}$,

   $$g(x) = x^\alpha L_g(x),\ x \in (0, 1],$$

   where $L_g : (0, 1] \to [0, \infty)$ is continuously differentiable, slowly varying at 0 and bounded away from 0. Moreover there exists a constant $C > 0$ such that the derivative L'_g of L_g satisfies

   $$|L'_g(x)| \leq C(1 + x^{-1}),\ x \in (0, 1].$$

2. The function g is continuously differentiable on $(0,\infty)$, with derivative g' that is ultimately monotonic and also satisfies $\int_1^\infty g'(x)^2 dx < \infty$.

Remark 18 Assumption 1 is the description of the similarity of $g(x)$ and x^α near zero. While assumption 2 is necessary for square integrability of $g(x)$. For a general BSS process assumption 2 isn't sufficient for the square integrability. Therefore a third condition is needed: For some $\beta \in (\infty, -\frac{1}{2})$,

$$g(x) = \mathcal{O}(x^\beta),\ x \to \infty.$$

Example 1 Kernel functions that satisfy the assumptions are for example [8, example 2.3 & 2.4]

- the *gamma kernel*

  $$g(x) = x^\alpha e^{-\gamma x},\ x \in (0,\infty),$$

  with $\alpha \in (-\frac{1}{2}, \frac{1}{2})\backslash\{0\}$ and $\gamma > 0$

- the *power-law kernel*

  $$g(x) = x^\alpha (1+x)^{\beta-\alpha},\ x \in (0,\infty)$$

  with $\alpha \in (-\frac{1}{2}, \frac{1}{2})\backslash\{0\}$ and $\beta \in (-\infty, -\frac{1}{2})$.

Now we are able to derive the hybrid scheme [cp. 8, section 2.3]. If we discretizise X_t given by 4.4.5 on the grid $G_t^n = \{t, t-\frac{1}{n}, t-\frac{2}{n}, \dots\}$ for $n \in \mathbb{N}$, we get

$$X_t = \sum_{k=1}^{\infty} \int_{t-\frac{k}{n}}^{t-\frac{k}{n}+\frac{1}{n}} g(t-s)\sigma_s dW_s.$$

Because we can't do numerical calculations up to ∞, we truncate the sum at $N_n \in \mathbb{N},\ N_n > 0$. N_n denotes our number of discretization steps. Further we keep the volatility process σ 4.4.5 constant in each discretization cell. This is reasonable if σ doesn't vary too much. Since σ will be constant in the rBergomi model, we don't face a problem with this assumption. Hence we get

$$X_t \approx \sum_{k=1}^{N_n} \sigma_{t-\frac{k}{n}} \int_{t-\frac{k}{n}}^{t-\frac{k}{n}+\frac{1}{n}} g(t-s) dW_s. \tag{7.2.1}$$

The hybrid scheme now splits the kernel function into two parts. If $k \le \kappa$, where $\kappa \in \mathbb{N}$ is small, i.e. in our simulation $\kappa \in [0,3]$, the kernel function gets approximated due to assumption 1

$$g(t-s) \approx (t-s)^\alpha L_g\left(\frac{k}{n}\right),\ t-s \in \left[\frac{k-1}{n}, \frac{k}{n}\right]\backslash\{0\}, \tag{7.2.2}$$

because the slowly varying function L_g varies less than the power function x^α near zero. If $k > \kappa$ we just approximate $g(x)$ directly

$$g(t-s) \approx g\left(\frac{b_k}{n}\right),\ t-s \in \left[\frac{k-1}{n}, \frac{k}{n}\right], \tag{7.2.3}$$

with $b_k \in [k-1, k]$.
Hence if we apply 7.2.2 and 7.2.3 to 7.2.1 we get the *hybrid scheme*

$$X_t \approx \sum_{k=1}^{\kappa} L_g\left(\frac{k}{n}\right) \sigma_{t-\frac{k}{n}} \int_{t-\frac{k}{n}}^{t-\frac{k}{n}+\frac{1}{n}} (t-s)^\alpha dW_s + \sum_{k=\kappa+1}^{N_n} g\left(\frac{b_k}{n}\right) \sigma_{t-\frac{k}{n}} \int_{t-\frac{k}{n}}^{t-\frac{k}{n}+\frac{1}{n}} dW_s.$$

Remark 19 $b = (b_k)_{k=\kappa+1}^{\infty}$ is a sequence of real numbers that only needs to satisfy $b_k \in [k-1, k]\backslash\{0\}$. But there is an optimal sequence $(b_k^\star)_{k=\kappa+1}^{\infty}$, that minimizes the asymptotic MSE [8].

Proposition 1 *Let $\alpha \in (-\frac{1}{2}, \frac{1}{2})\backslash\{0\}$ and $\kappa \geq 0$. Among all sequences $b = (b_k)_{k=\kappa+1}^{\infty}$ with $b_k \in [k-1, k]\backslash\{0\}$ for $k \geq \kappa+1$ the asymptotic MSE induced by the discretization is minimized by the sequence b^* given by*

$$b_k^* = \left(\frac{k^{\alpha+1} - (k-1)^{\alpha+1}}{\alpha+1}\right)^{\frac{1}{\alpha}},\ k \geq \kappa+1.$$

The proof can be found in [8, p. 941 Proposition 2.8].

7.2.1 Implementation of the hybrid scheme

The practical implementation is taken from [8, 3.1 and 3.3].

Remark 20 Following the notations of [8], we further use the parameter α in the rBergomi model instead of H. It holds, that

$$\alpha = H - \frac{1}{2}.$$

Because $H \in (0, \frac{1}{2})$ it follows, that $\alpha \in (-\frac{1}{2}, 0)$.

We want to use the hybrid scheme for the calculation of the rBergomi model. We do that by using the scheme for the calculation of Y_t, as defined below, from the spot variance price process

$$v_t := \xi_t^0 \exp\left(\eta \underbrace{\sqrt{2\alpha+1} \int_0^t (t-s)^\alpha dW_s}_{=:Y_t} - \frac{\eta^2}{2} t^{2\alpha+1}\right). \tag{7.2.4}$$

Note that Y_t is a TBSS process with kernel function $g(x) = x^\alpha$. Hence we can choose $L_g(x) = 1\ \forall x$ as slowly varying function.

At first we must show, that our function kernel g and L_g fulfil the assumptions 1 and 2 for the kernel function of the hybrid scheme. Assumption 1 is obvious fulfilled, because $g(x) = x^\alpha$ for $\alpha \in (-\frac{1}{2}, 0)$ and $L_g \equiv 1$ satisfies the desired conditions as well. For assumption 2 note, that $g'(x) = \alpha x^{\alpha-1}$ is continuous and strictly monotonic thus ultimately monotonic. Further it holds

$$\begin{aligned}\int_1^\infty g'(x)^2 dx &= \int_1^\infty (\alpha x^{\alpha-1})^2 dx = \alpha^2 \int_1^\infty x^{2(\alpha-1)} dx \\ &= \alpha^2 \left(\frac{1}{2(\alpha-1)} x^{2\alpha-1} \Big|_1^\infty \right) = -\frac{\alpha^2}{2(\alpha-1)} < \infty,\end{aligned}$$

because from $\alpha \in (-\frac{1}{2}, 0)$ follows that $2\alpha - 1 < 0$. Hence the kernel function fulfils the assumptions and we can use the hybrid scheme for our simulations.

Note that in the rBergomi model holds $\sigma \equiv 1$. Thus we obtain with $g(x) = x^\alpha$ and $L_g \equiv 1$ on the equidistant grid $\{0, \frac{1}{n}, \frac{2}{n}, \dots, \frac{nT}{n}\}$ the following scheme

$$Y^n_{\frac{i}{n}} = \sum_{k=1}^{\min(i,\kappa)} \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k}{n}+\frac{1}{n}} \left(\frac{i}{n} - s \right)^\alpha dW_s + \sum_{k=\kappa+1}^{i} \left(\frac{b^*_k}{n} \right)^\alpha \int_{\frac{i}{n}-\frac{k}{n}}^{\frac{i}{n}-\frac{k}{n}+\frac{1}{n}} dW_s \quad (7.2.5)$$

$$=: \sum_{k=1}^{\min(i,\kappa)} W^n_{i-k,k} + \sum_{k=\kappa+1}^{i} W^n_{i-k}. \quad (7.2.6)$$

For the simulation of Y we therefore first need to generate the random variables:

$$W^n_{i,j} := \int_{\frac{i}{n}}^{\frac{i+1}{n}} \left(\frac{i+j}{n} - s \right)^\alpha dW_s,$$

$$W^n_i := \int_{\frac{i}{n}}^{\frac{i+1}{n}} dW_s.$$

We do this by generating $\kappa + 1$ dimensional random vectors of the form

$$W^n_i := (W^n_i, W^n_{i,1}, \dots, W^n_{i,k}),\ i = -N_n, -N_n + 1, \dots, \lfloor nT \rfloor - 1. \quad (7.2.7)$$

In [8, section 3.1 & 4.3] it is shown that the vector (7.2.7) is i.i.d. according to a multivariate Gaussian distribution with mean zero and covariance matrix Σ given by

$$\begin{aligned}\Sigma_{1,1} &= \frac{1}{n}\\ \Sigma_{1,j} = \Sigma_{j,1} &= \frac{(j-1)^{\alpha+1} - (j-2)^{\alpha+1}}{(\alpha+1)n^{\alpha+1}}\\ \Sigma_{j,j} &= \frac{(j-1)^{2\alpha+1} - (j-2)^{2\alpha+1}}{(2\alpha+1)n^{2\alpha+1}}\end{aligned}$$

for $j = 2, \dots, \kappa+1$ and

$$\begin{aligned}\Sigma_{j,k} = \frac{1}{(\alpha+1)n^{2\alpha+1}} \Big(&(j-1)^{\alpha+1}(k-1)^{\alpha}{}_2F_1\left(-\alpha, 1, \alpha+2, \frac{j-1}{k-1}\right)\\ &-(j-2)^{\alpha+1}(k-2)^{\alpha}{}_2F_1\left(-\alpha, 1, \alpha+2, \frac{j-2}{k-2}\right)\Big) \quad (7.2.8)\end{aligned}$$

for $j, k = 2, \dots, \kappa+1$, such that $j < k$ and ${}_2F_1$ is the Gauss hyper-geometric function. If $k < j$ set $\Sigma_{j,k} = \Sigma_{k,j}$.
Thus we get the random vectors W_i^n, $i = -N_n, \dots, \lfloor nT \rfloor - 1$ by drawing independent samples from $\mathcal{N}_{\kappa+1}(0, \Sigma)$.

Before we continue we will state the proof of equation (7.2.8) from [8, section 4.3]. For that we use again Lemma 7.

Proof Let $\alpha \in (-\frac{1}{2}, \frac{1}{2})\backslash\{0\}$ and let $j, k = 2, \dots, \kappa+1$ be such that $j < k$. By the Itô isometry and by substituting $x = ns$, we have

$$\begin{aligned}\Sigma_{j,k} = \mathbb{E}[W^n_{0,j-1}W^n_{0,k-1}] &= \int_0^{\frac{1}{n}} \left(\frac{j-1}{n} - s\right)^{\alpha} \left(\frac{k-1}{n} - s\right)^{\alpha} ds\\ &= \frac{1}{n^{2\alpha}} \int_0^{\frac{1}{n}} (j-1-ns)^{\alpha}(k-1-ns)^{\alpha} ds\\ &= \frac{1}{n^{2\alpha+1}} \int_0^{1} (j-1-x)^{\alpha}(k-1-x)^{\alpha} dx.\end{aligned}$$

Now with the help of Lemma 7 we get

$$\begin{aligned}&\int_0^1 (j-1-x)^{\alpha}(k-1-x)^{\alpha} dx\\ &= \int_0^{j-1} (j-1-x)^{\alpha}(k-1-x)^{\alpha} dx - \int_0^{j-2} (j-2-x)^{\alpha}(k-2-x)^{\alpha} dx\\ &= \frac{1}{\alpha+1}\Big((j-1)^{\alpha+1}(k-1)^{\alpha}{}_2F_1\left(-\alpha, 1, \alpha+2, \frac{j-1}{k-1}\right)\\ &\quad -(j-2)^{\alpha+1}(k-2)^{\alpha}{}_2F_1\left(-\alpha, 1, \alpha+2, \frac{j-2}{k-2}\right)\Big).\end{aligned}$$

The Lemma is applicable to both integrals as $0 < j-1 < k-1$ and $0 \leq j-2 < k-2$. Combining both equations concludes the proof. $\square$

With the random vectors $W_i^n,\ i = -N_n, \ldots, \lfloor nT \rfloor - 1$ we can calculate (7.2.5). Because we take $\kappa = 0, \ldots, 3$ in our simulation the first sum doesn't have a worth mentioning influence on the computational effort. But the second sum can have a notable influence as the number of terms increase as $n \to \infty$. Thus we use a convolution for the second sum [8, p. 947]. Note that

$$\sum_{k=1}^{N_n} \Gamma_k \Xi_{i-k} = (\Gamma \star \Xi)_i, \tag{7.2.9}$$

holds with

$$\Gamma_k := \begin{cases} 0, & k = 1, \ldots, \kappa \\ \left(\frac{b_k^*}{n}\right)^\alpha, & k = \kappa+1, \kappa+2, \ldots, N_n \end{cases} \tag{7.2.10}$$

$$\Xi_k := W_k^n,\ k = -N_n, -N_n+1, \ldots, \lfloor nT \rfloor - 1. \tag{7.2.11}$$

Where $(\Gamma \star \Xi)$ is the discrete convolution which can be efficiently computed by a fast Fourier transform (FFT) algorithm.
Therefore let's introduce the Fourier transform [p. 6, Chapter 5]

Definition 13 (Fourier transform) Let $f \in L^1(\mathbb{R})$, then define its *Fourier transform* by

$$\tilde{f}(u) = \int_{\mathbb{R}} e^{iux} f(x) dx.$$

Theorem 5 *Let $f, g \in L^1(\mathbb{R})$ and $h = (f \star g)$. Then $h \in L^1(\mathbb{R})$ and it holds*

$$\tilde{h}(u) = \tilde{f}(u)\tilde{g}(u).$$

Now we can proof 5

Proof It holds that

$$(f \star g) = \int_{\mathbb{R}} f(y) g(x-y) dy.$$

First we need to show that $h \in L^1(\mathbb{R})$. With Fubini's theorem [24, theorem 6.10.5] we get

$$\begin{aligned} \|f \star g\|_{L^1} &= \int_{\mathbb{R}} \left| \int_{\mathbb{R}} f(y) g(x-y) dy \right| dx \\ &\leq \int_{\mathbb{R}} \int_{\mathbb{R}} |f(y) g(x-y)|\, dy dx \quad = \int_{\mathbb{R}} |f(y)|\, dy \int_{\mathbb{R}} |g(x)|\, dx = \|f\|_{L^1} \|g\|_{L^1} < \infty. \end{aligned}$$

Therefore we get $h \in L^1(\mathbb{R})$. Now using Fubini's theorem once again we have that

$$\begin{aligned}\widetilde{(f \star g)}(u) &= \int_{\mathbb{R}} e^{iux}(f \star g)(x)dx \\ &= \int_{\mathbb{R}} e^{iux} \int_{\mathbb{R}} f(y)g(x-y)dydx \\ &= \int_{\mathbb{R}} e^{iuy} f(y)dy \int_{\mathbb{R}} e^{iu(x-y)} g(x-y)dx = \tilde{f}(u)\tilde{g}(u).\end{aligned}$$

□

For the description of the method for calculating the convolution with the Fourier transform we further need the inverse Fourier transform, which is defined as follows.

Definition 14 Let $g \in L^1(\mathbb{R})$, then the *inverse Fourier transform* of g is defined as

$$\breve{g} = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-iux} g(x)dx.$$

Theorem 6 (Inversion Theorem) *Let $f, \tilde{f} \in L^1(\mathbb{R})$, then it holds for every $x \in \mathbb{R}$:*

$$f(x) = \breve{\tilde{f}}(x).$$

A proof of the theorem can be found in [6, Chapter 5].

Now the algorithm for calculating of a convolution $(f \star g)$ with the Fourier transform works as follows:

1. Calculate the Fourier transform $\tilde{f}$ and $\tilde{g}$ of f and g.
2. Multiply $\tilde{f}$ and $\tilde{g}$.
3. Calculate the inverse Fourier transform of the product $\tilde{f}\tilde{g}$. The inverse Fourier transform is then the result of the convolution.

8 Implementation

Before we start to look at our simulation results, we first want to give an overview of the programming language and some further packages we used for our implementation.
As a programming language we have chosen Python 3.6 with the anaconda distribution. As normally for scientific computing with Python we mainly used the NumPy package for our computations, but since we wanted to receive a high number N for our discretization steps and Monte Carlo paths M further use of other packages was necessary.
All simulations were done on a laptop with 16GB RAM and an INTEL i7-4720 with 2.60 GHz, 4 cores and 8 threads.
Because our simulation results get better if we increase N and M we wanted to increase those values as far as possible. Therefore we had to optimize the usage of the RAM, to be able to work with higher N's, write our results to the hard-drive, while we do a simulation, to increase M and increase the speed, so we could run our simulations with still executable run times.
For the optimization we mainly used the hints given in the presentation [2].
For the improvement of the general RAM usage in the calculations we used the *NumExpr* package [11]. *NumExpr* is a package designed for fast vector and matrix calculations, which also uses less memory than *NumPy*. While *NumPy* creates temporary arrays for calculations, *NumExpr* avoids the creation of temporary arrays. Hence it uses less memory, which allowed us to increase N. Furthermore it's extremely easy to use with the *numexpr.evaluate* function. For more information please look at the documentation under [11].
We used the package for all bigger array calculations with the level of optimization set to moderate, so we won't lose accuracy. The use of NumExpr made it possible to extend N to $2^{14} = 16384$ in the hybrid scheme.
Let us now focus on problem of writing our data on hard-drive. *NumPy* offers an easy solution for writing data on a hard-drive, but the problem is, that it's not possible to write in an existing file. Instead each run of the simulation would need to create a new file with data, which would end in a huge amount of files. Therefore we used *PyTables* [36]. *PyTables* is a package which creates HDF5 files (Herarchical Data Format, see [17]), which can store multiple arrays at once and are fast in terms of reading and writing big amounts of data.
In our simulations we created a file with an predefined array, which could store our simulation data, and than after each run we wrote the data to a part of the array. Thus after the simulation we received one file, which stored all the necessary informations for our analysis.
This allows us to set M to an arbitrary number, which is only limited by the

time we want to wait for our results.

So in the end our program works as follows:

1. Start the simulation for $2^{12} = 4096$ Monte Carlo paths. The value is chosen, so that the memory can handle the calculations for $N = 16384$ discretization steps with $\kappa = 3$.

2. After the simulation is done, write the results for S_T on the hard-drive.

3. Repeat the simulation for a given number of desired repetitions. Through setting the number of repetitions, we could get our arbitrary number of Monte Carlo paths.

Remark 21 Because the total number of paths of our Monte Carlo simulation is determined by the number of repetitions, we give our results in term of the repetitions and not the total numbers. One receives the total number of runs through $4096 * repetitions$. The repetitions were chosen in a way one would roughly receive 10,000; 50,000; 100,000; 200,000; 1,000,000 respectively 4,000,000 Monte Carlo paths. In the following tables the actual numbers of Monte Carlo paths to the repetitions are given:

Repetitions	total numbers
3	12,288
13	53,248
25	102,400
49	200,704
250	1,024,000
1000	4,096,000

Table 2: Total number of MC runs for the repetitions

We further used *PyFFTW*, python package for calling the C-library FFTW, which is again fast C-library for the calculation of Fast Fourier Transforms (FFT) [18], for the calculation of the convolution (7.2.9).
The convolution is then done by calculating the FFT of our variables, multiply the fast Fourier transforms of both variables and calculate the inverse Fourier transform of the product.
Instead of using a FFTW we could also just have used *numpy.convolve* but this is much slower. For $N = 8,192$ and $\kappa = 3$ then generation of Y (7.2.4) with *numpy.convolve* takes 60 seconds and for $N = 16,384$ with *PyFFTW* we only need around 35 seconds. Hence we doubled N and only needed

around half the runtime.

Remark 22 R. McCrickerd, the co-author of [29], made a Python implementation of the hybrid Scheme for the rBergomi model for $\kappa = 1$. He published his code on GitHub [30]. We used his code for debugging our implementation.

For the calculation of the covariance matrix of the exact scheme we used Cython [12], which enables the programmer to easily write C extensions for Python and the declaration of variables, which greatly improves the speed of loops. Since we calculate the entries of the covariance matrix by going through the matrix with for-loops, we benefit greatly from the loop speed-up.
After we calculate the covariance matrix, we once save it on the hard-drive, because the calculation for $N = 2^{10}$ and 2^{11} is quite time consuming. Thus we could use the created covariance matrices again, if we changed our code for the simulation of the rBergomi model. The simulation of the exact scheme then follows the same steps as the simulation of the hybrid scheme, beside at the beginning we read in the covariance matrix.

The analysis of the obtained simulation is then done in a different Python program, which only reads in the results, calculates our values of interest and creates the plots.

Another package which has no influence on the computation but comes in handy for the data analysis is *pandas* [35]. It enables the creation of tables with the help of data frames and contains a function called *to_latex()*, which directly writes a data frame as a table in LaTeX-code.

8.1 Influence of different random number computations

As we already mentioned we wanted to get a good result in the 4^{th} decimal place in our simulations. But as we found in our simulations besides the algorithm the functions or packages we choose for our random number generation or functions in general also have an impact on our results in the 4^{th} decimal place. Normally this wouldn't be so significant, but since we want to determine for which κ the hybrid scheme gives us the best result, it becomes important for us. So we also need to keep in mind the accuracy respectively influence of the functions we use in our program. As we will see below two

random number generators, which are both well tested and return the same type of random variables, can make a different in the result of our simulation. It is important to note, that we are talking about minor changes. The values we get in the end differ maximal around 0.0001. Furthermore to really examine the differences of the Cholesky decomposition versus the direct function call in 8.1.2 we would have needed to set the same seed for the random number generator, which we haven't done. Nevertheless we see some minor differences and tendencies in our results we want to address. Furthermore we want to raise awareness, that even well tested methods and functions lead to slightly different results in the computations. The worst part is, that we can't say which of the solutions we obtained is the correct one. They are all plausible and thus can be the correct solutions. Hence we see that even with 4 million Monte Carlo paths we only get an idea for the 4^{th} decimal place but not necessary the correct solution.

Note that all values we talk about in the next section are calculated with 4 million Monte Carlo paths.

8.1.1 Random number generators

We used during our simulations two different random number generators.

1. *numpy.random*, which is the standard choice for scientific computing with Python.

2. Intel® Math Kernel Library random number generator, which is developed by Intel® and optimized for Intel® processors. Original it is a C-library but a Python wrapper *mkl_random* exists.

Both are well tested and are used, so that we can trust both, that they give us correct results. They both feature functions for the creation of standard normal and multivariate normal random variables for a given mean and covariance matrix.
Table 3 contains the prices we have calculated with the hybrid scheme and $\kappa = 1, \ldots, 3$ with the different random number generators.

Simulation	price	- 95% confidence interval	-95% confidence interval
$\kappa = 1$ *NumPy*	0.07913874	0.07904271	0.07923478
$\kappa = 1$ *MKL*	0.07922693	0.07913080	0.07930778
$\kappa = 2$ *NumPy*	0.07923478	0.07918203	0.07937501
$\kappa = 2$ *MKL*	0.07924308	0.07914660	0.07933955
$\kappa = 3$ *NumPy*	0.07937501	0.07927862	0.07947224
$\kappa = 3$ *MKL*	0.07940386	0.07930708	0.07950065

Table 3: Prices of the hybrid scheme with different random number generators

If we look at the changes between the two generators, we notice that *mkl_random* leads to slightly higher results than *numpy.random*. The next table contains the differences of the prices and confidence intervals. A positive number means, that the price with *mkl_random* was higher than the one with *numpy.random* and vice versa.

As noted before the differences are really small and it is not possible to say,

Simulation	price	- 95% confidence interval	-95% confidence interval
$\kappa = 1$	0.000088	0.000088	0.000073
$\kappa = 2$	0.000008	-0.000035	-0.000035
$\kappa = 3$	0.000029	0.000028	0.000028

Table 4: Difference of prices of the hybrid scheme with different random number generators

which result would be better. Nevertheless in general *mkl_random* seems to give us slightly higher prices than *numpy.random*.

8.1.2 Cholesky decomposition versus direct function call

For our simulations we need to get multivariate normal random variables with covariance matrix Σ. One possibility is to calculate the Cholesky decomposition of Σ, create $\mathcal{N}(0,1)$ random variables and calculate the dot product of both. Another option is to use *numpy.random.multivariate_normal* with Σ, which then returns multivariate normal random variables with covariance matrix Σ.

The first method is actually faster than the second. For the exact scheme with $N = 4,096$ *numpy.random.multivariate_normal* needs up to 186 seconds to generate the random variables. The Cholesky decomposition needs only around 6-7 seconds and 30 seconds for the calculation of the decomposition, which then is held in memory and used for the repetitions. For

the Cholesky decomposition we also used functions provided by *NumPy*, *numpy.linalg.cholesky* and the *.dot* method for *NumPy* arrays.
But again both methods lead to slightly different results. For the exact scheme for $N = 2,048$ we get the following values for the two different methods.

Simulation method	price	- 95% confidence interval	-95% confidence interval
numpy multivariate	0.07907168	0.07897608	0.07916727
Cholesky	0.07893926	0.07884377	0.07903475

Table 5: Difference of prices of the exact scheme with different multivariate random number calculations

If we use the Cholesky decomposition the value for the price is about 0.00013242 smaller, which is not much. But if we look at the confidence intervals, we see that both values aren't contained in the 95% confidence intervals of one another.
Sadly we can't say which of all the values is the correct one. Since *NumPy* is the standard for Python we decided to use the values we obtained by using the function *numpy.random.multivariate_normal* for our analysis. Unfortunately this gives us only maximal $2,048$ discretization steps for the exact scheme instead of $4,096$ for the Cholesky decomposition. Besides we also used *numpy.random.multivariate_normal* for the hybrid scheme. Hence if we have a certain effect by choosing *numpy.random.multivariate_normal* for the generation of the random numbers we at least have the effect carried through all our simulations and thus the values are consistent.

9 Weak error analysis of the rBergomi model - exact scheme

In this section we will analyse our simulation results for the exact scheme of the rBergomi model. At first we will look at the order of convergence from the ROC and the EOC and in the second subsection we will take a look at the results from the difference monte carlo scheme.

Remark 23 All simulations were done for the rBergomi model with $H = 0.07$ respectively $\alpha = -0.43$, $\eta = 1.9$, $\xi = 0.0235^2$ and $\rho = -0.9$. As function we used the call price $(S_T - k)^+$ with $r = 0$, $S_0 = 1$ and $k = 1$.

Note that for the exact scheme we only give our results for $M = 4,096,000$ paths.

9.1 Rate of convergence

N	weak error	EOC
2	0.035372	
4	0.018002	0.974419
8	0.009436	0.931930
16	0.004938	0.934357
32	0.002557	0.949548
64	0.001262	1.018217
128	0.000596	1.083423
256	0.000293	1.022460
512	0.000223	0.393577
1024	0.000080	1.487901

Table 6: Weak error and EOC of the exact scheme

In table 6 we see the weak error for each number of discretization steps N and $4,096,000$ Monte Carlo paths. As reference solution we used our value for $N = 2,048$ discretization steps.
One way to obtain an idea of the order of convergence is to look at the EOC. The values are also given in table 6. We see that besides the values for $N = 512$ and 1024 all values are in the range of 1 ± 0.07. This clearly would indicate an order of convergence of 1. Further we see, that the EOC gets larger for larger N. We assume that this is, because we get closer to our reference solution. The values for 512 and 1024 seem to be statistical outliers. From 256 to 512 steps the error decreases slower while from 512 to

1024 the error decreases a bit faster. This can be caused by the integration error. As the number of discretization steps increases, also the variance increases. Hence the integration error increases and thus we get worse results for higher N. Since all other values lead to the assumption that the rate of convergence is 1, we still would assume that this is the correct order of convergence and the last two values are caused by the increasing variance.
In the following we calculate the order of convergence from the slope of the linear regression line of the weak errors form table 6 in a log-log plot. Figure 3 contains the log-log plot we obtained and the calculated regression line. The lighter area around the plot is the 95% confidence interval. We further included a dashed line with slope 1. Thus we see, that the regression line is parallel to the dashed line. Therefore we conclude that the exact scheme convergences with order 1. Further the calculated order of convergence is $\gamma = 0.96187$, which we would also indicate as order of convergence 1.

In table 7 we give the calculated values of the rate of convergence. We have also included our values for 49 and 250 repetitions. Hence we can see that order of convergence gets closer to 1 from 49 to 250 repetitions, while from 250 to 1000 repetitions we only see a minor change in the values.

Repetitions	γ	C
49	1.122044	0.083266
250	0.966396	0.068569
1000	0.961872	0.069156

Table 7: C and γ of the rate of convergence of the exact scheme

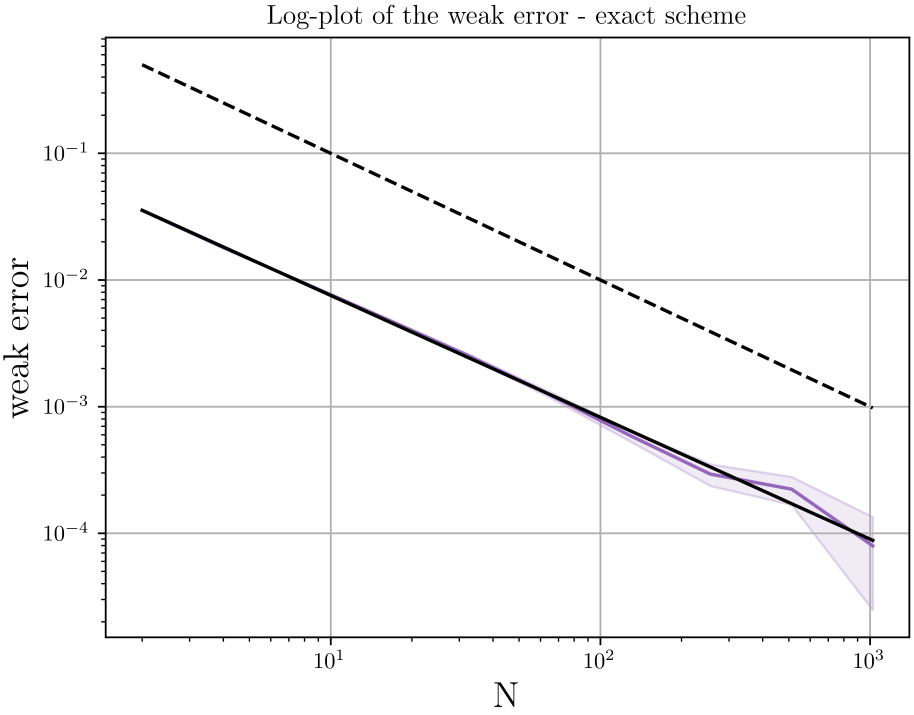

Figure 3: Log-log plot of the weak error - exact scheme

Note that the lower part of the confidence interval has a bigger peak in the figure. The peaks up and down have the same size, but due to the log-log plot the lower part seems to be bigger. Further we see that the integration error increases as we increase the number of discretization steps.

Since the regression line and the weak error in the log-log plot overlap, we also give the log-log plot of the weak error without the regression line. Then we nicely see even without the regression line, that the weak error is almost parallel to our dashed orientation line.

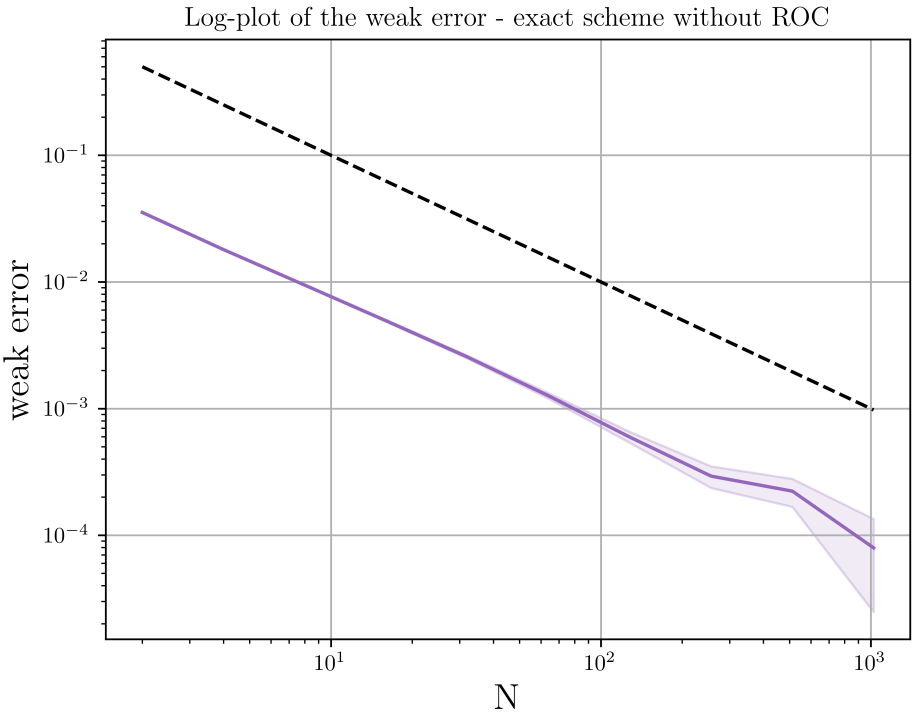

Figure 4: Log-log plot of the weak error - exact scheme without regression line

9.2 Difference Monte Carlo scheme

Also we already followed from the previous section, that the scheme convergences with order 1, we are still going to use the difference Monte Carlo scheme to verify our assumption.
The weak errors and the EOCs we calculated with the difference Monte Carlo scheme are given in table 8.

N	weak error	EOC
2	0.017369	
4	0.008566	1.019818
8	0.004498	0.929262
16	0.002381	0.917864
32	0.001294	0.879278
64	0.000667	0.957346
128	0.000302	1.140169
256	0.000070	2.110941
512	0.000144	-1.036732
1024	0.000080	0.851928

Table 8: Weak error and EOC of the difference Monte Carlo scheme - exact scheme

We again see that the EOC is in the range of 1 besides the values for $N = 256$ and 512. If we look at the figure 5, we see that we have a peak at $N = 256$. Hence the error decreases more from $N = 128$ to $N = 256$, which leads to an higher EOC_{256} of 2, and then increases again from $N = 256$ to 512, which gives us a negative EOC_{512}. The peak is caused by the increasing variance of our estimator for higher N. We don't see a contradiction in the values for $N = 256$ and 512 to our assumption, that the order of convergence is 1.

As second approach we calculate the ROC of the difference Monte Carlo scheme. By this the peak at $N = 256$ shouldn't have an impact on the overall order of convergence we obtain from the ROC.
In figure 5 we see the obtained weak error and the calculated ROC. The figure again contains a dashed line with slope 1 and the 95% confidence interval.

We see in the plot that the regression line is almost parallel to our guide line. It is a little flatter but not so much, that we would conclude the order of convergence differs from 1. In fact $\gamma = 0.915948$, which we still interpret as 1.

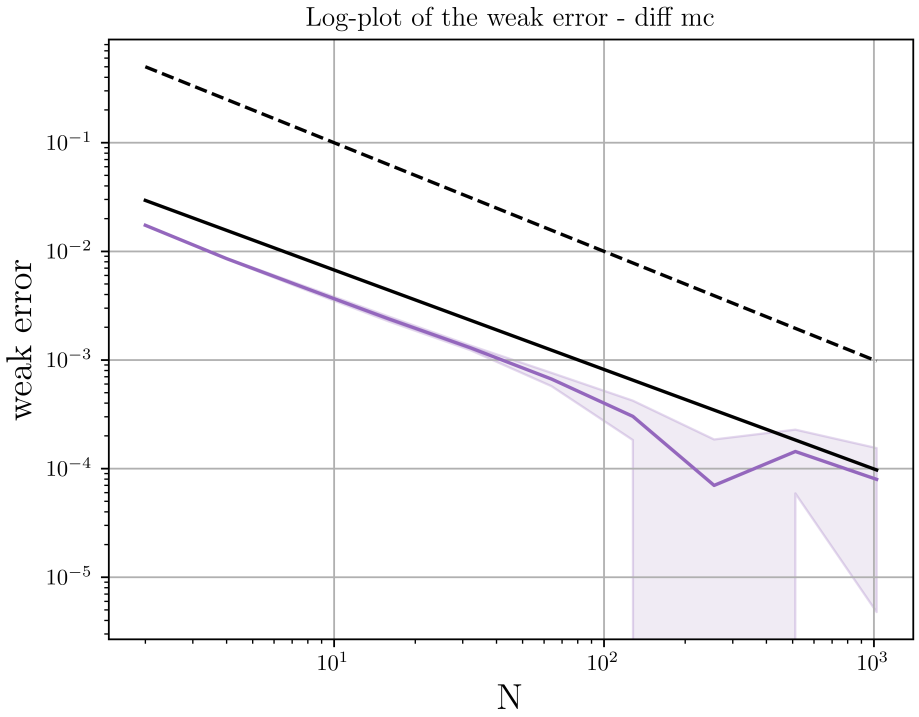

Figure 5: Log-log plot of the weak error of the difference Monte Carlo scheme - exact scheme

Table 9 contains the parameters of the ROC. We see again that γ gets closer to 1 for more repetitions, but surprisingly 250 repetitions give us a better result than 1000 repetitions.

Repetitions	γ	C
49	0.816930	0.039176
250	0.988884	0.0685686
1000	0.915948	0.055521

Table 9: C and γ of the rate of convergence of the difference Monte Carlo scheme - exact scheme

10 Weak error analysis of the rBergomi model - hybrid scheme

For the analysis of the weak error of the hybrid scheme we use the same methods as for the exact scheme but this time we take our result from the exact scheme as our reference solution for calculating the ROC.
Before we start with the analysis of the weak error, we will take a look at the implied volatility. Because M. Bennedsen et al. [8] described the characteristics for the hybrid scheme for the implied volatility, we can use the implied volatility to check our implementation of the hybrid scheme.

10.1 Implied volatility

Before we start with the actual analysis of the weak error we shortly take a look at the implied volatility. So far the literature on the rBergomi and the hybrid scheme concentrated on the implied volatility. As a test if our simulation is working correctly for $\kappa = 0, \dots 3$ we look if we get the same results as M. Bennedsen et al [8] for the implied volatility. They found in their research, that the implied volatilities for $\kappa = 1, \dots, 3$ aren't distinguishable, while $\kappa = 0$ can capture the shape, but not the level [8, p. 952] of the implied volatility.

Note that we only calculate the implied volatility for the option price of a call option for $K = 1$. Thus we don't obtain the typical implied volatility smile. Still we see the same characteristics.
Note that we only calculated the exact scheme for N up to 2048. Hence for $N > 2048$ we continued the value from $N = 2048$.
Further we used the results we got for the hybrid scheme with 250 repetitions, since we didn't calculate a value for $\kappa = 0$ for 1000 repetitions. For the implied volatility of the exact solution we used our simulation results with 1000 repetitions.

As we see in figure 6 $\kappa = 0$ can't capture the level of the implied volatility while the exact scheme and the hybrid scheme with $\kappa = 1, \dots, 3$ deliver pretty similar results. For $N < 5000$ we see, that the values for different κ's are still different. This is the case because the slightly different rates of convergences for the κ's, as we will see later. If $N > 7500$ we can't distinguish the lines of $\kappa = 2$ and 3 anymore, while the line of $\kappa = 1$ is a little but lower. Interestingly $\kappa = 1$ gives us the best approximation of the implied volatility of the exact scheme. We would expect that $\kappa = 3$ gives us the best

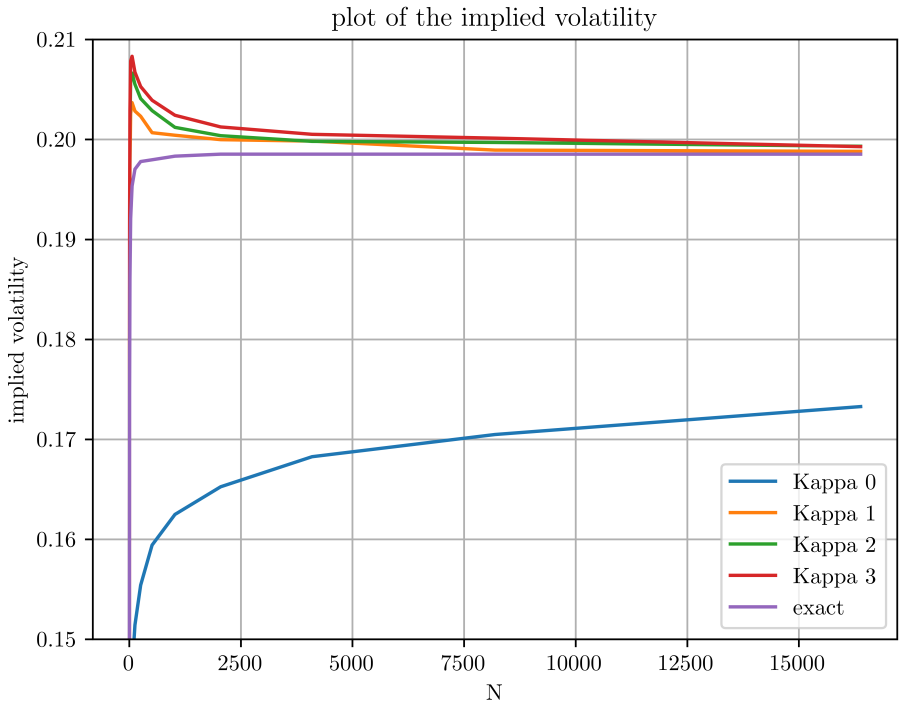

Figure 6: Plot of the implied volatility - hybrid scheme, 250 repetitions

approximation of the exact solution.

Note that the actual differences between the 4 values are really small, in the case of 250 repetitions we have an difference of maximal 0.00047 and for 1000 repetitions of 0.00077, see table 10 for the actual values. So we are talking about differences in the 4^{th} decimal place.

Repetitions	exact	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
250	0.19880983	0.17325214	0.19856715	0.19923156	0.19928445
1000	0.19852885		0.1986978	0.1990499	0.19929402

Table 10: Implied volatility of the hybrid and exact scheme

10.2 Rate of convergence

Let us now take a look at the weak error of the hybrid scheme. As reference solution we took the price we obtained by the exact scheme for 4 million Monte Carlo paths.
In figure 7 we plotted the weak error for $\kappa = 0, \ldots, 3$. The lighter areas around the graphs are again the 95% confidence intervals. For the plots we used for $\kappa = 1, \ldots, 3$ the results we got for 4 million paths, while for $\kappa = 0$ we only used 1 million paths, since as we made our runs with 4 million paths it was already clear that $\kappa = 0$ wouldn't convergence to the correct solution. Hence we didn't simulate $\kappa = 0$ for 4 million paths.

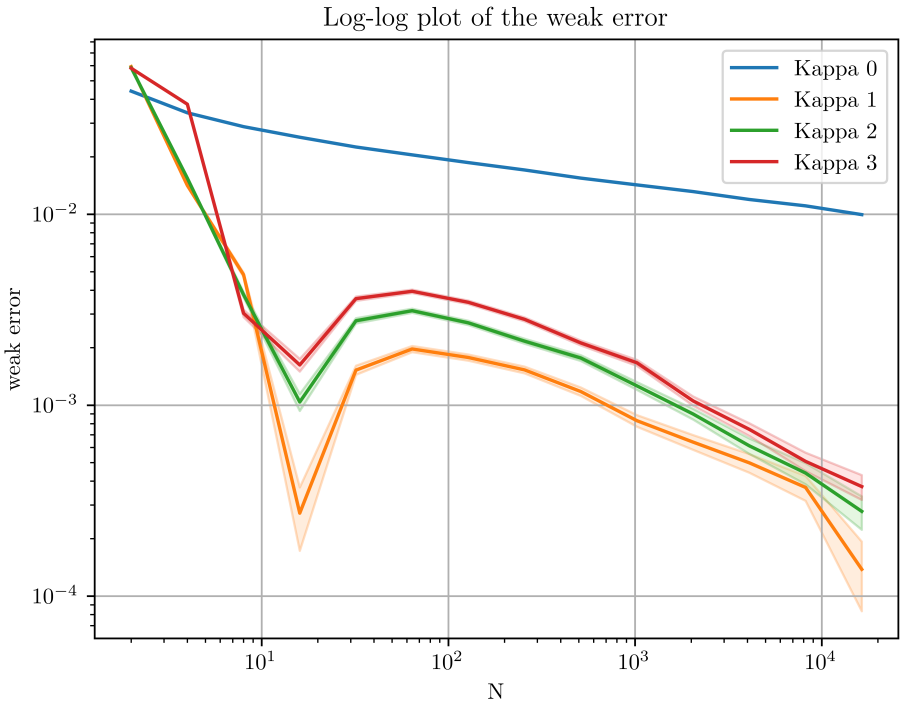

Figure 7: Log-log plot of the weak error of the hybrid scheme

As we see in the figure $\kappa = 0$ doesn't convergence to our correct solution. We don't get a lower weak error than 10^{-2}. So $\kappa = 1$ can neither capture the right level of the implied volatilty nor convergences to the correct value in the weak error sense.
The graphs for $\kappa = 1, \ldots, 3$ are pretty close to each other and they all have

a similar shape, as we would expect. Further the difference between $\kappa = 1$ and $\kappa = 2$ is bigger than between $\kappa = 2$ and $\kappa = 3$. But for increasing N the differences get smaller. Furthermore we see that the three graphs all have a peak around 10^1. This can be due to statistical errors.
Nevertheless we get two interesting results:

1. $\kappa = 1$ gives us the lowest weak error, which is surprising since we would expect that the weak error gets lower for higher κ.
   Note that the differences are really small. The difference for the weak error from $\kappa = 1$ to $\kappa = 2$ is only 0.00014 and from $\kappa = 2$ to $\kappa = 3$ 0.000097 for $N = 16,384$, as we can see in table 12, which contains the weak errors of figure 7.
   To be able to do this differentiation we needed the 4 million Monte Carlo paths for our calculations because for $200,000$, 1 and 4 million Monte Carlo paths we got the following prices with the exact scheme:

Repetitions	price	- 95% confidence interval	+ 95% confidence interval
49	0.0793304	0.07889785	0.07976294
250	0.07918322	0.0789920	0.07937444
1000	0.07907168	0.07897608	0.07916727

Table 11: Prices obtained by the exact scheme for different repetitions

   If we would have used the value for 49 repetitions 0.0793 as our reference solution, then the hybrid scheme with $\kappa = 3$ would have given us the smallest weak error, followed by $\kappa = 2$ and $\kappa = 1$. Our simulation result for 250 repetitions 0.07918 already led to $\kappa = 1$ giving us the smallest error and $\kappa = 3$ the highest. But since for 250 repetitions our 95% confidence interval was still so big, that it also included 0.0793 the opposite result would still be possible. Thus we had to even further increase our number of paths. First for 4 million paths we got a small enough 95% confidence interval so that we could say the result is more in the range of $0.079 - 0.0791$ than 0.0793 and hence $\kappa = 1$ gives us a smaller weak error than $\kappa = 3$.
   Also through the 4 million paths we were able to distinguish the values for $\kappa = 1, \ldots, 3$. As we see in figure 7 the confidence intervals of the hybrid scheme are nearly disjoint for 4 million paths, which is necessary to really determine if the hybrid scheme convergences to different results for $\kappa = 1, \ldots, 3$.

2. If we increase κ from 1 to 3 the peak around 10^1 gets smaller, see figure

7.

	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
2	0.044146	0.059330	0.058623	0.058194
4	0.033997	0.014157	0.015447	0.037706
8	0.028766	0.004824	0.003792	0.003030
16	0.025315	0.000272	0.001040	0.001625
32	0.022504	0.001527	0.002769	0.003615
64	0.020470	0.001970	0.003126	0.003951
128	0.018647	0.001778	0.002700	0.003459
256	0.017060	0.001531	0.002165	0.002817
512	0.015467	0.001180	0.001767	0.002120
1024	0.014248	0.000830	0.001262	0.001668
2048	0.013147	0.000641	0.000899	0.001053
4096	0.011951	0.000499	0.000613	0.000747
8192	0.011069	0.000371	0.000441	0.000508
16384	0.009958	0.000138	0.000278	0.000375

Table 12: Weak error of the hybrid scheme

In the following we will again take a look at the EOC, which is contained in the following table.

As expected $\kappa = 0$ has the lowest EOC of around 0.15. For $\kappa = 1, \ldots, 3$ we notice that the values for $N < 128$ are first quite high before they become negative. This caused by the peak at $N = 16$. Before we reach the peak the weak error drops quite fast and hence the EOC is quite high. After we reached the peak at $N = 16$ the EOC is negative because the error increases again. First for $N \geq 1,024$ the EOC even outs at 0.5. Interestingly the EOC for $\kappa = 2$ from $N = 1,024$ to $8,192$ is higher than for $\kappa = 1$. Also $\kappa = 3$ has a higher EOC for those N than $\kappa = 1$ with the exception of $N = 1,024$. For $\kappa = 2$ and $\kappa = 3$ we can't say, that one is superior than the other. Sometimes $\kappa = 2$ has a higher EOC and sometimes $\kappa = 3$.
Thus from the EOC would follow that for $N = 1,024$ to $8,192$ the hybrid scheme for $\kappa = 2, 3$ actually convergences a little bit faster than for $\kappa = 1$.

Further we conclude from the EOC, that the order of convergence for $\kappa = 1, \ldots, 3$ is around 0.5.

N	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
4	0.376887	2.067282	1.924150	0.626097
8	0.241010	1.553211	2.026423	3.637517
16	0.184367	4.148007	1.866550	0.898832
32	0.169860	-2.488629	-1.412853	-1.153753
64	0.136651	-0.367448	-0.175379	-0.127996
128	0.134546	0.148414	0.211615	0.191907
256	0.128311	0.215354	0.318543	0.296039
512	0.141465	0.376289	0.292720	0.410386
1024	0.118422	0.507755	0.485603	0.345927
2048	0.115979	0.372782	0.489719	0.663752
4096	0.137597	0.360561	0.553354	0.494670
8192	0.110632	0.427718	0.473119	0.556615
16384	0.152549	1.424618	0.666843	0.438321

Table 13: EOC of the hybrid scheme

To get an idea about which of the three κ's gives us the overall fastest convergence we calculate the ROC. In figure 8 we have plotted the weak errors with the calculated ROC. In the plots for $\kappa = 1, \ldots, 3$ we have included a dashed reference line with slope -0.5.

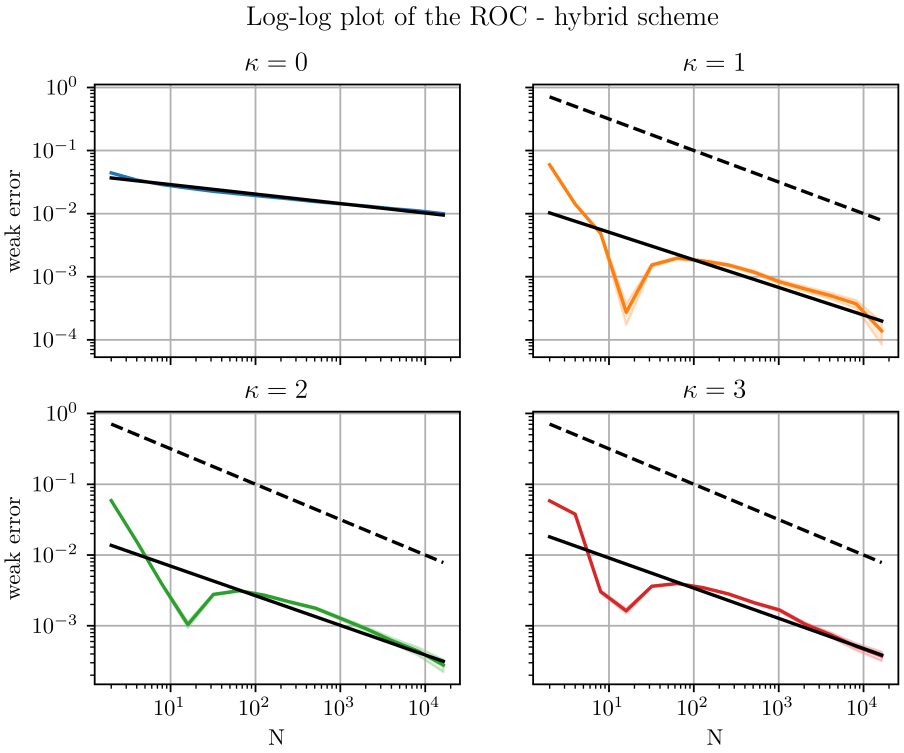

Figure 8: Log-log plot of the ROC and the weak error of the hybrid scheme

As we can see, the regression lines are almost parallel to the dashed lines. They are a little bit flatter but we would still interpret this as an order of convergence of 0.5.

Repetitions	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
3	0.148364	0.585926	0.448855	0.425848
13	0.150009	0.529767	0.593518	0.437380
25	0.149575	0.541393	0.401195	0.409133
49	0.150654	0.485843	0.410037	0.446729
250	0.150318	0.443937	0.409999	0.411016
1000		0.437665	0.418418	0.426604

Table 14: Order of convergence γ of the hybrid scheme

Repetitions	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
3	0.040547	0.019574	0.019175	0.027871
13	0.040661	0.025143	0.03583	0.025877
25	0.040756	0.019266	0.016563	0.02431
49	0.040566	0.014755	0.017811	0.025769
250	0.040721	0.014357	0.018367	0.023295
1000		0.013882	0.018258	0.024246

Table 15: Constant C of the ROC of the hybrid scheme

Table 14 contains the calculated values for the order of convergence. The values are more in the range of 0.45. Note that the values for $\kappa = 1$ decrease as the repetitions increase. Besides the constants C of the ROC, see table 15, are also higher for 3 to 25 repetitions and fluctuate more. Thereafter they are around 0.014.
On the other hand the values for $\kappa = 2$ and $\kappa = 3$ are more consistent for the different number of repetitions.
To further analyse the fluctuations we plotted the weak errors we obtained for all number of Monte Carlo paths for $\kappa = 1, \ldots, 3$ in figure 9.
As we can see for $\kappa = 1$ the scheme fluctuates much more for fewer Monte Carlo paths, than the other two and the scheme for $\kappa = 3$ fluctuates less than for $\kappa = 2$. So the scheme itself seems to convergence smoother for higher κ but nevertheless $\kappa = 1$ gave us slightly better results.

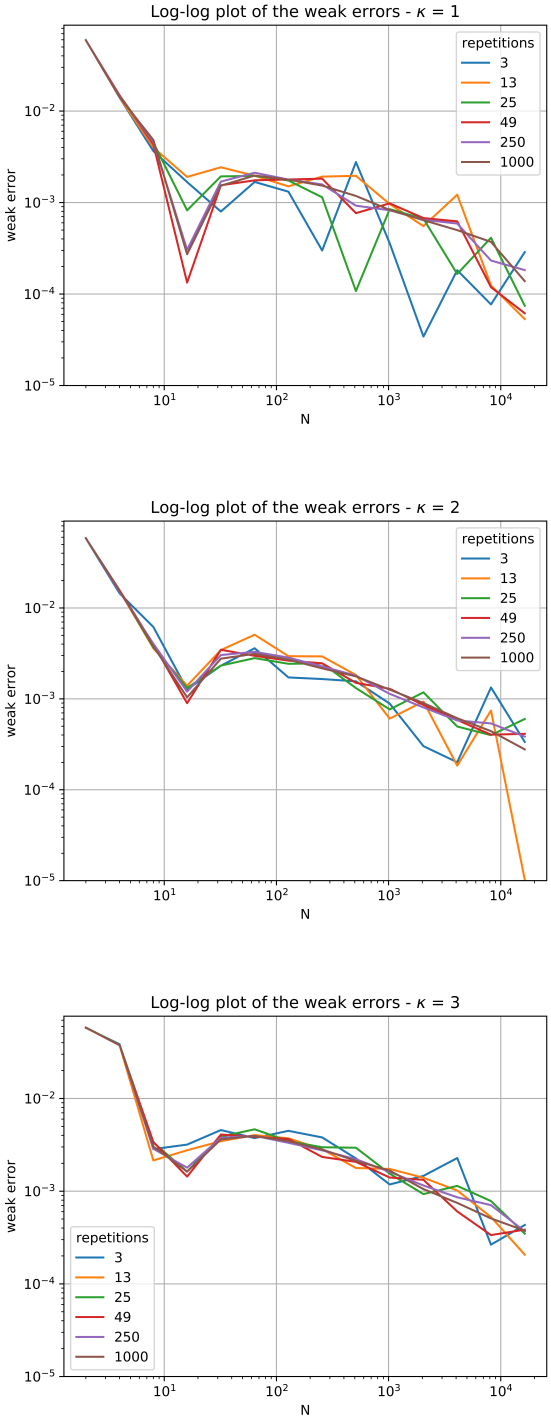

Figure 9: Log-log plots of the weak errors of the hybrid scheme for the different repetitions

If we take again a look at the values for the order of convergence from table 14, we further see that for 1000 repetitions the hybrid scheme with $\kappa = 1$ gave us the best result, followed by $\kappa = 3$ and then $\kappa = 2$. If we only look at the order of the convergence, we get the wrong impression, that $\kappa = 3$ gives us a lower weak error. But we also need to consider the constant C of the ROC and $C_{\kappa=3} > C_{\kappa=2}$. Hence if we consider this and look at the actual plot of the regression lines, we see that $\kappa = 2$ gives us the better results for the weak error.
Note that of course the regression lines of $\kappa = 2$ and 3 will cross, but therefore we would need to set $N \approx 10^{15}$.

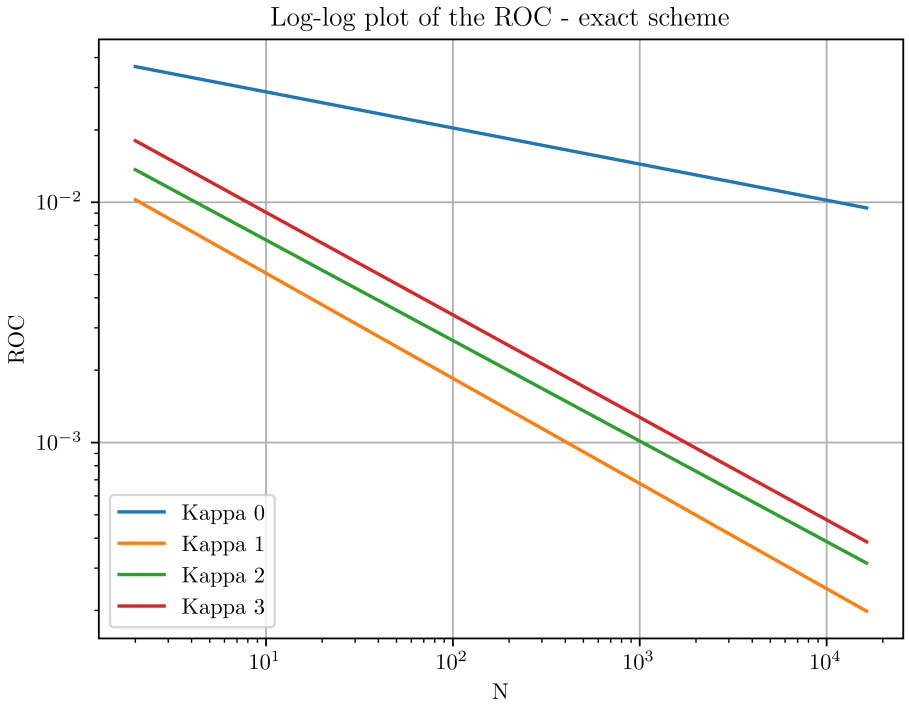

Figure 10: Log-log plot of the ROC of the hybrid scheme

10.3 Difference Monte Carlo scheme

In this subsection we will look at the results from the difference Monte Carlo scheme. In the following figure we plotted the weak errors we got from the hybrid scheme. Note that we left out the values for $\kappa = 0$ to get a clearer image.

We see in the plot, that all three values are again pretty similar. Up to

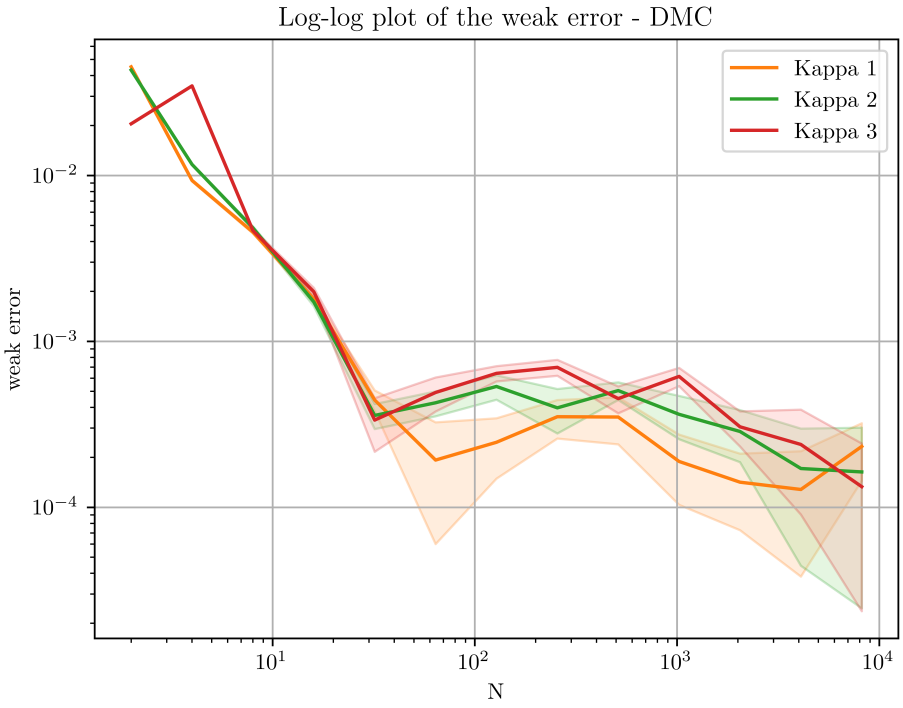

Figure 11: Log-log plot of the weak error of the difference Monte Carlo scheme - hybrid scheme

$N = 32$ the values are nearly identical. For $N = 64$ to 8192 $\kappa = 1$ gives us again the best result followed by $\kappa = 2$ and $\kappa = 3$. For $N = 16,384$ the difference Monte Carlo scheme returns a smaller weak error for $\kappa = 3$ and $\kappa = 2$ than for $\kappa = 1$.
This is due to the fact, that the difference Monte Carlo scheme approximates the weak error for each given κ, i.e. it approximates the weak error

$$\mathbb{E}[f(\hat{X}_\kappa) - f(X_\kappa))], \ \kappa = 0, \ldots, 3,$$

where $\hat{X}_\kappa$ is the result of the approximation and X_κ is the correct solution for given κ. Hence we get a smaller error form the difference Monte Carlo

scheme for $\kappa = 2, 30$ for $N = 16384$, because the differences from $N = 8192$ to $N = 16384$ is smaller than for $\kappa = 1$.
At first let's look at the EOC we get.

N	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
4	0.956410	2.275097	1.889253	-0.759111
8	0.599902	1.035868	1.270463	2.897190
16	0.295493	1.339061	1.482705	1.225625
32	0.467400	2.022088	2.272260	2.569053
64	0.158012	1.201657	-0.253257	-0.552937
128	0.199926	-0.355903	-0.326605	-0.382877
256	-0.006007	-0.512056	0.428057	-0.120313
512	0.386626	0.006381	-0.345424	0.626076
1024	0.147284	0.889494	0.475368	-0.444583
2048	-0.119955	0.415018	0.343375	1.009030
4096	0.438984	0.146197	0.741609	0.353552
8192	-0.332227	-0.862540	0.068477	0.845552

Table 16: EOC of the difference Monte Carlo scheme - hybrid scheme

For the difference Monte Carlo scheme, we don't get usable results for the EOC. The values differ too much for all N and κ to identify any kind of tendencies.

Let's therefore jump ahead and look at the values for the ROC.

This time none of the three values for κ give us a better result than the other. For different number of Monte Carlo paths other values for κ are better. But the values are still in the around 0.5, which at least confirms this result.

Repetitions	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
3	0.148364	0.585926	0.448855	0.425848
13	0.38322	0.451724	0.428127	0.49757
25	0.252669	0.423171	0.518772	0.513468
49	0.24554	0.573964	0.712326	0.60403
250	0.240853	0.666834	0.614413	0.515476
1000	0.240853	0.583139	0.554311	0.543588

Table 17: Order of convergence γ from the difference Monte Carlo scheme - hybrid scheme

Repetitions	Kappa = 0	Kappa = 1	Kappa = 2	Kappa = 3
3	0.040547	0.019574	0.019175	0.027871
13	0.013321	0.011979	0.014332	0.014524
25	0.007756	0.008826	0.014006	0.018999
49	0.007627	0.012191	0.035175	0.023257
250	0.00756	0.024206	0.023789	0.017427
1000	0.00756	0.017344	0.019109	0.020807

Table 18: Constant C of the ROC from the difference Monte Carlo scheme - hybrid scheme

If we look at the constant we obtained from the ROC, $\kappa = 1$ has for all number of repetitions besides 250 the smallest constant. Since the order of convergence doesn't differ that much, we would therefore conclude that we still get the lowest weak error for $\kappa = 1$.

If we look at the plot of the ROC for $N = 1000$, we get overall the same result for the ROC.
Hence we see this as an acknowledgement of our result from section 10.2.

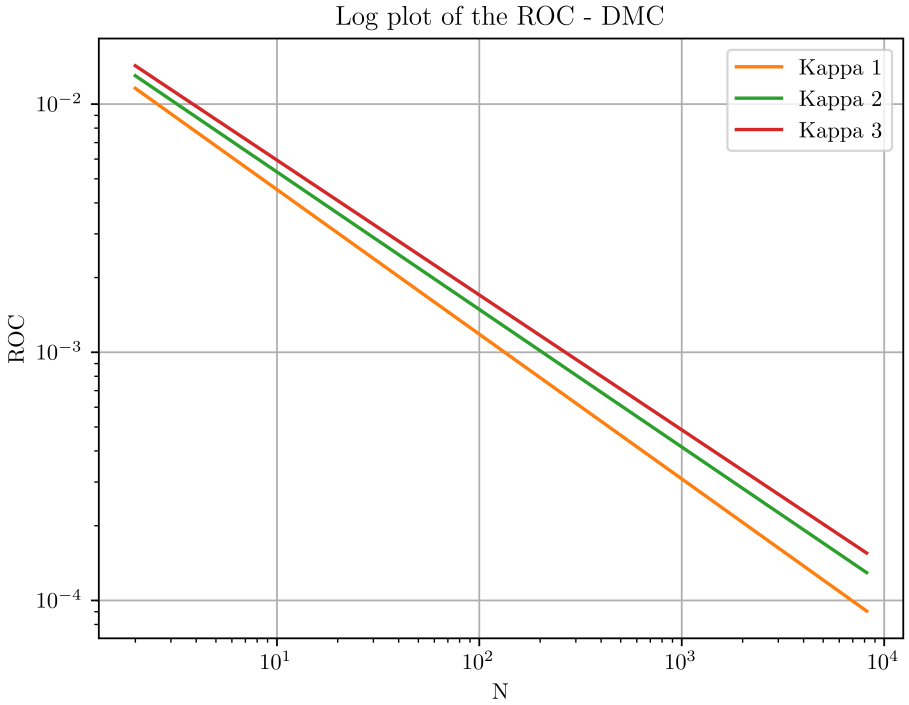

Figure 12: Log-log plot of the ROC from the difference Monte Carlo scheme - hybrid scheme

10.4 Analysis of the hybrid scheme for different parameters

To back up our results of the order of convergence 0.5 we did two simulations of the hybrid scheme for other parameters. In the following we will only give our results from the ROC since this method gave us before the best results. Note that for those simulations we only used 49 repetitions respectively $200,000$ Monte Carlo paths. Therefore we won't have the needed precision in our results, to determine which κ gives us the best result, because the 95% confidence intervals are to big. They are not disjoint, which would be necessary for a clear differentiation.
Nevertheless the order of convergences we obtain are in the range of $0.4-0.68$, which we still interpret as order of convergence 0.5. So we are able to confirm the order of convergence of the previous analysis.

1. Simulation results for $H = 0.07$ respectively $\alpha = -0.43$, $\eta = 1.9$, $\xi = 0.0235^2$ and $\rho = 0$.
   As reference solution for the weak error we used two values:

   - a value we obtained by C. Bayer from an exact simulation for 512 discretization steps and $200,000$ Monte Carlo paths
   - and a value from the hybrid scheme for $\kappa = 1$ with 1 million Monte Carlo paths.

   Note that we decided to use the value from the hybrid scheme for $\kappa = 1$ since this value gave us before the best results.

	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
γ	0.159038	0.504207	0.684046	0.525907
C	0.045888	0.016115	0.021692	0.017171

Table 19: Parameters of the ROC from reference solution by C. Bayer, with 49 repetitions

	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
γ	0.157433	0.475378	0.538295	0.643895
C	0.046015	0.013792	0.013526	0.024234

Table 20: Parameters of the ROC from reference solution from the hybrid scheme, with 49 repetitions

	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
γ	0.280862	0.374946	0.467502	0.484094
C	0.010171	0.011015	0.012732	0.016105

Table 21: Parameters of the ROC from the difference Monte Carlo scheme, with 49 repetitions

2. Simulation results for $H = 0.02$ respectively $\alpha = -0.48$, $\eta = 0.4$, $\xi = 0.1$ and $\rho = -0.7$.

   Here we used as reference solution a value from the hybrid scheme for 1 million Monte Carlo paths and $\kappa = 3$. Note that we used $\kappa = 3$ because we calculated those values, before we assumed, that $\kappa = 1$ gives us the best results in terms of the weak error.

	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
γ	0.248079	0.579727	0.640176	0.445279
C	0.020085	0.018536	0.02044	0.012653

Table 22: Parameters from the ROC, 49 repetitions

	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$
γ	0.402932	0.40941	0.573971	0.54415
C	0.009854	0.011711	0.018033	0.019624

Table 23: Parameters form the ROC from the difference Monte Carlo scheme, 49 repetitions

11 Conclusion

We were able to determine per simulation an order of convergence for the exact and the hybrid scheme for the rBergomi model. The exact scheme convergences with order 1 and the hybrid scheme for $\kappa > 0$ with order 0.5.

Further we were able to determine that the hybrid scheme for the rBergomi gives us slightly different results for $\kappa = 1, \dots, 3$ and that those differences are of the magnitude 10^{-4}. Nevertheless we saw that $\kappa = 1$ gave us better results in terms of the weak error than $\kappa = 2$ or $\kappa = 3$.

This is a bit surprising but convenient, because the implementation of the hybrid scheme for $\kappa = 1$ is easier than for $\kappa > 1$, since for $\kappa = 1$ the covariance matrix is given by

$$\begin{aligned}
\Sigma_{1,1} &= \frac{1}{n} \\
\Sigma_{1,2} = \Sigma_{2,1} &= \frac{1}{(\alpha+1)n^{\alpha+1}} \\
\Sigma_{2,2} &= \frac{1}{(2\alpha+1)n^{2\alpha+1}},
\end{aligned}$$

thus we don't have to work with the ${}_2F_1$ function and further we need less time for the computation as we can see in the following figure.

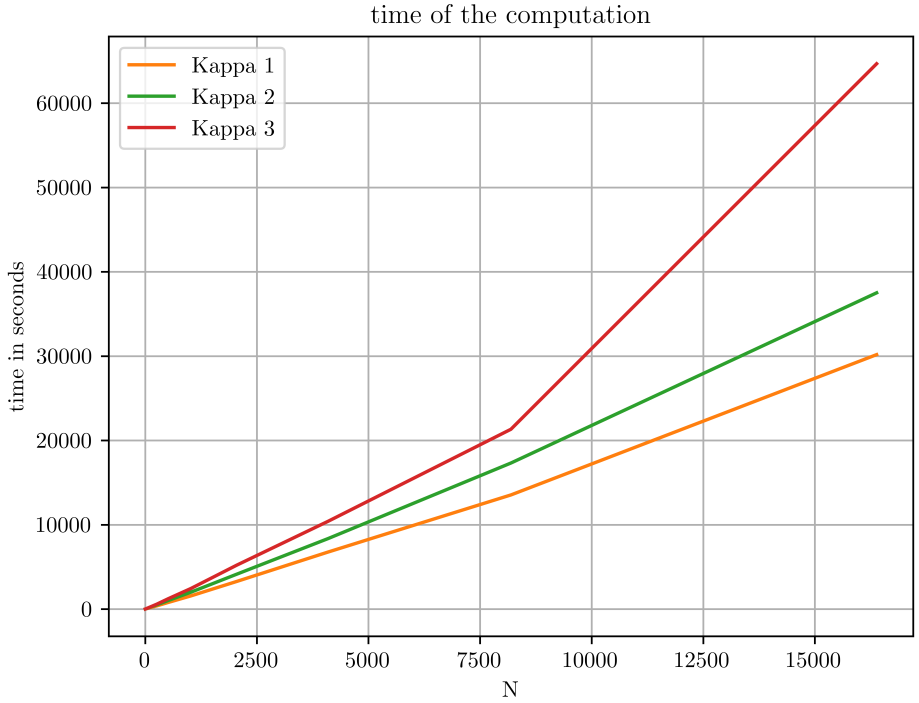

Figure 13: Time of the computation of the hybrid scheme with NumPy as random number generator for 4 million paths

12 Appendix

12.1 Python code for the exact scheme

12.1.1 Cython code for the covariance matrix

```
cimport cython
from libc.math cimport sqrt
import numpy as np
cimport numpy as np
import mpmath

DTYPE = np.double
ctypedef np.double_t DTYPE_t

def G(double x, double gamma,double H):
return 2*H*((1./(1.-gamma))*x**(-1.*gamma) + (gamma/(1-gamma))*x
    **(-1.*(1.+gamma))*(1./(2.-gamma))*mpmath.hyp2f1(1.,1.+gamma
    ,3.-gamma,x**(-1.)))


cpdef cov_matrix_rBergomi(int steps,double a, double p, np.
    ndarray[double] t):
cdef double H = a + 0.5
cdef double gamma = -1.*a
cdef double DH = sqrt(2*H)/(H + 0.5)
cdef np.ndarray cov = np.zeros((2*steps,2*steps), dtype = np.
    double)
cdef int i, j
for i in range(2*steps):
if i < steps:
cov[i,i] = t[i+1]**(2*H)
#first loop for W from 0 to steps
for j in range(i+1,steps):
cov[i,j] =  t[i+1]**(2*H) * G(t[j+1]/t[i+1],gamma, H)
#second loop for Z from steps to 2*steps
for j in range(max(steps,i+1),2*steps):
cov[i,j] = p*DH*((t[i+1])**(H+0.5) - (t[i+1] - np.minimum(t[i
    +1],t[j+1-steps]))**(H+0.5))
else:
cov[i,i] = t[i+1-steps]
for j in range(i,2*steps):
cov[i,j] = min(t[i+1-steps],t[j+1 -steps])
#loop for the lower part of the covariance matrix
for j in range(0,i):
cov[i,j] = cov[j,i]
return cov
```

The function for writing the covariance matrix to the hard drive is the following.

```
import cov_rBergomi_3
import time
import numpy as np
import tables

def cov_matrix_rBergomi_file(m,a,p):
for i in range(1,m+1):
steps= 2**i
shape_carray = (2*steps,2*steps)
t = np.linspace(0, T, T * (2**i) + 1)
fileName = f"****.hdf5"
hdf5 = tables.open_file(fileName, mode='w')
filters = tables.Filters(complevel=5, complib='blosc')
simulation_data = hdf5.create_carray(hdf5.root, 'cov', tables.
    Float64Atom(), shape_carray, filters=filters)
simulation_data[:,:] = cov_rBergomi_3.cov_matrix_rBergomi(steps,
    a, p, t)
hdf5.close()

if __name__ == '__main__':
# setting the maximum number of steps to 2**m
m = 11
# alpha
a = -0.43
# rho
p = -0.9
cov_matrix_rBergomi_file(11,a,p)
```

12.1.2 Code for the exact scheme

```
import math
import numexpr as ne
import numpy as np
import time
import tables
import mpmath
import cov_rBergomi
from numpy.random import multivariate_normal


def gen_V_rBergomi(W, a, xi, eta, steps,t,N):
W_part = W[:,:steps]
t_part = t[1:]
return ne.evaluate('xi * exp( eta * W_part - 0.5 * eta**2 *
    t_part**( 2 * a + 1))', optimization='moderate')

def gen_Z_rBergomi(W, N, steps):
Z = np.zeros((N,steps))
Z[:,0] = W[:,steps]
for i in range(1,steps):
Z[:,i] = W[:,steps + i] - W[:,steps + i-1]
return Z

def gen_S_rBergomi(V, Z, S0, dt, N, steps):
"""
Calculates the value of the underlying S in the rBergomi model.

Input parameters:
V - the spot variance process of the rBergomi model,
see page 951 of the hybrid scheme paper
output of the function gen_V(Y, t, alpha, eta, xi)
Z - a standard brownin motion with Z = p * W + sqrt(1 - p^2 ) *
    W_orth
output of the function gen_Z(W,W_orth,p)
S0 - the price of the underlying at t = 0
dt -  the size of each step (stepzise)

Output:
the value of the underlying S in the rBergomi model for each t
"""

V_part = V[:,:-1]
Z_part = Z[:,1:]
integral = np.cumsum(ne.evaluate('sqrt(V_part) * Z_part - 0.5 *
    V_part * dt', optimization='moderate'), axis=1)
S = np.empty((N, steps))
S[:, 0] = S0
```

```
S[:, 1:] = ne.evaluate('S0*exp(integral)', optimization='
    moderate')
return S


def run_simulation_rBergomi(N, m, T,  S0, a, eta, xi, p, repeats
    ):
parameters = np.array([N*repeats, 2**m, m, T, 1/(2**m), S0, 0, a
    , eta,xi, p])
fileName = f"*****.hdf5"
shape_carray = (N*repeats, m)
atom = tables.Float64Atom()
hdf5 = tables.open_file(fileName, mode='w')
filters = tables.Filters(complevel=5, complib='blosc')
simulation_data = hdf5.create_carray(hdf5.root, 'simulation',
    atom, shape_carray, filters=filters)
parameters_data = hdf5.create_carray(hdf5.root, 'parameters',
    atom,  (1,11), filters=filters)
time_date = hdf5.create_carray(hdf5.root, 'time', atom, (m,4),
    filters = filters)
parameters_data[0,:] = parameters
for i in range(1,m+1):
start = time.time()
steps = 2**i
dt = 1.0/steps
t = np.linspace(0, T, T * 2**i + 1)
start1 = time.time()
cov_matrix = tables.open_file(f"****.hdf5", mode='r')
for r in range(repeats):
print('steps = ', steps, ' noch ', repeats - r, ' Wiederholungen
    ')
W_rBergomi = multivariate_normal(np.zeros(2*steps), cov_matrix.
    root.cov[:,:],N, check_valid='ignore')
V_rBergomi = gen_V_rBergomi(W_rBergomi,a,xi,eta,steps,t,N)
Z_rBergomi = gen_Z_rBergomi(W_rBergomi,N,steps)
simulation_data[N*r:N*(r+1), i-1] = gen_S_rBergomi(V_rBergomi,
    Z_rBergomi,S0,dt,N,steps)[:,steps-1]
time_date[i-1,:] = np.array([N*repeats,steps, k, time.time()-
    start])
cov_matrix.close()
hdf5.close()
```

12.2 Python code for the hybrid scheme

```
import math
import numexpr as ne
import numpy as np
import time
import tables
import pyfftw
import mpmath
from numpy.random import multivariate_normal

def gen_Z(W, W_orth, p, N, steps):
"""
gen_Z generates Z = p * W + sqrt( 1 - p^2 ) * W_orth

Input parameters:
Number W - brownian motion with cov matrix.
Number W_orth - standard brownian motion independent of W.
Number p - the correlation parameter.
N - the total number of simulations.
steps - the number of steps in each simulation.

Output:
Z - the corralated brownain motion
"""
return ne.evaluate('p*W + sqrt(1-p**2)*W_orth', optimization='
    moderate')

def b_opt(k, a):
"""
Calculates the optimal b for the hybrid scheme.
The optimal b minimizes the error of the hybrid scheme.

Input parameters:
k - the indice of the b
a - the parameter alpha element of (-0.5,0.5) \ {0}

Ouput:
the optimal b for the given k and alpha (a)

Defined by
Bennedsen, Lunde, Pakkanen
Hybrid scheme for Brownian semistationary processes
Finance Stoch (2017) 21:931-965
page 941
"""
return ((k ** (a + 1) - (k - 1) ** (a + 1)) / (a + 1)) ** (1 / a
    )
```

```
def cov_matrix(k, a, n):
"""
Calculates the covariance matrix for the hybrid scheme,
as definied on page 946.

Input parameters:
k - the parameter kappa of the hybrid scheme. (page 945)
a - the parameter alpha of the hybrid scheme.
n - the parameter n of the hybrid scheme, steps per time unit.
n is equal to steps.

Output:
A covariance matrix.

Defined by
Bennedsen, Lunde, Pakkanen
Hybrid scheme for Brownian semistationary processes
Finance Stoch (2017) 21:931-965
page 945-946
"""
cov = np.zeros((k + 1, k + 1))
cov[0, 0] = 1. / n
for j in range(1, k + 1):
cov[0, j] = (j ** (a + 1) - (j - 1) ** (a + 1)) / ((a + 1) * n
    ** (a + 1))
cov[j, 0] = cov[0, j]
cov[j, j] = (j ** (2 * a + 1) - (j - 1) ** (2 * a + 1)) / ((2 *
    a + 1) * n ** (2 * a + 1))
for m in range(1, k + 1):
if (j < m):
cov[j, m] = (1 / ((a + 1) * n ** (2 * a + 1))) * (
j ** (a + 1) * m ** a * mpmath.hyp2f1(-a, 1, a + 2, (j * 1.0) /
    (m * 1.0)) - (j - 1) ** (
a + 1).real * (m - 1) ** a * mpmath.hyp2f1(-a, 1, a + 2, (j *
    1.0 - 1) / (m * 1.0 - 1)).real)
if (j > m and m != 0):
# need to swap j,m because of the loop iterrations
cov[j, m] = cov[m, j]
return cov

def gen_W(k, a, N, steps):
"""
Generates the multivariate normal distributed random variables
    needed for the
hybrid scheme. Therefore the random variables are created with
    the
covariance matrix given by the function cov_matrix.
```

```
Input parameters:
k - the parameter kappa of the hybrid scheme.
a - the parameter alpha of the hybrid scheme.
N - the number of total simulations.
steps, the number of steps in each simulation.

Output:
N(0,cov_matrix) multivariate normal variables.
"""
return multivariate_normal(np.zeros(k + 1), cov_matrix(k, a,
    steps), (N, 2*steps), check_valid='ignore')

def gen_W_orth(N, steps, dt):
"""
Generates a standard Brownian motion independent of W,
which is needed for the calculation of Z in the function gen_Z.

Input parameters:
N - the total number of simulations.
steps - the number of steps in each simulation.
dt - the size of each individual step (stepsize).

Output:
Standard Brownian motion
"""
return math.sqrt(dt)*np.random.randn(N, steps)


def gen_Y(kappa, W, a, N, steps, qmc):
"""
Calculate Y of the hybrid scheme for the rBergomi model
as on page 951. Y is defined on page 944 and 946
in "Hybrid scheme for Brownian semistationary processes".

Input parameters:
kappa, the parameter kappa of the hybrid scheme.
W - multivariate normal distributed random variables
with covariance matrix cov_matrix.
a - the parameter alpha of the rBergomi scheme.
N - the total number of simulations.
steps - the number of steps in each simulation.

Output:
Y as defined on page 951 for the rBergomi model.

Defined by
Bennedsen, Lunde, Pakkanen
Hybrid scheme for Brownian semistationary processes
Finance Stoch (2017) 21:931-965
```

```
"""

G = np.zeros(steps)
for i in range(kappa, steps):
G[i] = (b_opt(i+1, a) / steps) ** a

X1 = np.zeros((N,steps))
for i in range(steps):
for r in range(1,k+1):
X1[:,i] += W[:,i + steps - r, r]

pyfftw.interfaces.cache.enable()
nthreads = multiprocessing.cpu_count()
Xi = W[:,:steps,0]
shape = 2*steps + 1
fft_G_obj = pyfftw.builders.fft(G, n = shape, threads=nthreads,
    overwrite_input=True, planner_effort='FFTW_ESTIMATE',
    auto_align_input = False, auto_contiguous = False)
fft_W_obj = pyfftw.builders.fft(Xi,n= shape, threads=nthreads,
    overwrite_input=True, planner_effort='FFTW_ESTIMATE' ,
    auto_align_input = False, auto_contiguous = False)
ifft_obj = pyfftw.builders.ifft(fft_W_obj.get_output_array(),n=
    shape, threads=nthreads,overwrite_input=True, planner_effort=
    'FFTW_ESTIMATE',auto_align_input = False, auto_contiguous =
    False)
fft_padded_G = fft_G_obj(G)
fft_padded_W = fft_W_obj(Xi)
convolution = ifft_obj(fft_padded_G * fft_padded_W)[:,:steps]
convolution = convolution.real
convolution[:,:kappa] = 0

return ne.evaluate('sqrt(2.*a+1.)*(X1 + convolution)',
    optimization='moderate')

def gen_V(Y, t, alpha, eta, xi):
"""
Calculate the spot variance process v of the rBergomi model,
see page 951 of the hybrid scheme paper for the formula.

Input parameters:
Y - the parameter calculated by the hybrid scheme, see page 951,
output of the function Y(kappa,W,a,eta,N,steps)
t - t element of [0,T]
alpha - the parameter alpha of the rBergomi model
eta - the parameter eta of the rBergomi model -- look up what
    eta stands for
xi - the parameter xi of the rBergomi model -- look up what xi
    stands for
```

```
    Output:
    v as defined on page 951 hybrid scheme paper
    """
    t_part = t[1:]
    return ne.evaluate('xi * exp( eta * Y - 0.5 * eta**2 * t_part**(
        2 * alpha + 1))', optimization='moderate')

def gen_S(V, Z, S0, dt, N, steps):
    """
    Calculates the value of the underlying S in the rBergomi model.

    Input parameters:
    V - the spot variance process of the rBergomi model,
    see page 951 of the hybrid scheme paper
    output of the function gen_V(Y, t, alpha, eta, xi)
    Z - a standard brownin motion with Z = p * W + sqrt(1 - p^2 ) *
        W_orth
    output of the function gen_Z(W,W_orth,p)
    S0 - the price of the underlying at t = 0
    dt -  the size of each step (stepzise)

    Output:
    the value of the underlying S in the rBergomi model for each t
    """
    V_part = V[:,:-1]
    Z_part = Z[:,1:]
    integral = np.cumsum(ne.evaluate('sqrt(V_part) * Z_part - 0.5 *
        V_part * dt', optimization='moderate'), axis=1)
    S = np.zeros((N, steps))
    S[:, 0] = S0
    S[:, 1:] = ne.evaluate('S0*exp(integral)', optimization='
        moderate')
    return S

def run_simulation(N, steps, T, k, S0, a, eta, xi, p, m):
    """
    Runs the simulation of S and returns the values for ST.

    Input parameters:
    N - number of simulations
    steps - numbers of steps in the grid
    dt - size of the grid steps
    T - maturity
    k - parameter kappa of the hybrid scheme
    S0 - the value of the underlying at time 0
    alpha - the parameter alpha of the rBergomi model
    eta - the parameter eta of the rBergomi model -- look up what
        eta stands for
```

```
xi - the parameter xi of the rBergomi model -- look up what xi
    stands for
p - the correlation of W and W_orth, for the generation of Z
qmc - boolean, if True, then a qmc run is made, if False a
    normal
mc simulation is done

Output:
ST values for the underlying.
"""
dt = 1.0 / steps
t = np.linspace(0, T, T * steps + 1)
W = gen_W(k, a, N, steps)
W_orth = gen_W_orth(N, steps, dt)
Z = gen_Z(W[:, :steps, 0], W_orth, p, N, steps)
Y = gen_Y(k, W, a, N, steps, qmc)
V = gen_V(Y, t, a, eta, xi)
return  gen_S(V, Z, S0, dt, N, steps)[:,steps-1]


if __name__ == '__main__':
# initializing the values for the simulation
# number of simulations per repetitions
N = 2**12
# setting the maximum number of steps to 2**m
m = 14
# S0 the start price of the underlying
S0 = 1
# kappa
k = 3
# alpha
a = -0.43
# eta
eta = 1.9
# xi
xi = 0.235 ** 2
# rho the correlation for Z
p = -0.9
#set number of repeats

for repeats in [r]:
#save the parameters of the simulation in an array called
    parameters
parameters = np.array([N*repeats, steps, m,
T, dt, S0, k, a, eta,
xi, p])
#set desired path, where the simulation resutls shoud get saved
#set values for the hdf file
fileName = f"*******"
```

```
shape_carray = (N*repeats, m)
atom = tables.Float64Atom()
hdf5 = tables.open_file(fileName, mode='w')
filters = tables.Filters(complevel=5, complib='blosc')
#array where the simulation data gets saved
simulation_data = hdf5.create_carray(hdf5.root, 'simulation',
    atom, shape_carray, filters=filters)
#array where the parameters get saved
parameters_data = hdf5.create_carray(hdf5.root, 'parameters',
    atom, (1,11), filters=filters)
#array where the time information gets saved
time_date = hdf5.create_carray(hdf5.root, 'time', atom, (m,4),
    filters = filters)
parameters_data[0,:] = parameters
for i in range(1,m+1):
start2 = time.time()
for r in range(repeats):
print('steps = ', 2**i, ' still ', repeats - r, ' repeats', 'rho
    = ', p, 'kappa = ', k)
simulation_data[N*r:N*(r+1), i-1] = run_simulation(N,2**i,T,k,S0
    ,a,eta,xi,p,m)
time_date[i-1,:] = np.array([N*repeats,2**i, k, time.time()-
    start2])
hdf5.close()
```

13 Bibliograhpie

[1] F. Alted. *Large Data Analysis with Python.* G-Node, November 24th, 2010. Munich, Germany. `http://www.pytables.org/other_material.html`, viewed on the 4^{th} March 2018.

[2] O. E. Barndorff-Nielsen and J. Schmiegel. *Ambit processes: with applications to turbulence and tumour growth.* Stochastic Analysis and Applications, Abel Symp., vol. 2, pp. 93 - 124. Springer, Heidelberg (2007)

[3] O. E. Barndorff-Nielsen and J. Schmiegel. *Brownian semistationary processes and volatility/intermittency.* Radon Series on Computational and Applied Mathematics, Vol. 8, pp. 1 - 25. Walter de Gruyter, Berlin (2009).

[4] C. Bayer, P. Friz and J. Gatheral. *Pricing under rough volatility.* Quantitative Finance, Vol. 16, No. 6, 887-904, 2016.

[5] C. Bayer, P. Friz, A. Gulisashvili, B. Horvath, B. Stemper. *Short-time near-the-money skew in rough fractional volatility models.* WIAS preprint, Berlin, 2017.

[6] C. Bayer and A. Papapantoleon. *Computational Finance.* Lecture notes, TU Berlin, 2014.

[7] K. Benhenni. *Approximation integrals of stochastic processes: Extensions.* J. Appl. Probab., 35(4):843-855, 1998.

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

[9] L. Bergomi. *Smile dynamics II.* Risk October, pp. 67-73, 2005.

[10] P. Cheridito, H. Kawaguchi and M. Maejima. *Fractional Ornstein-Uhlenbeck processes.* Electronic Journal of Probability, Vol. 8 (2003) Paper no. 3, pages 1 - 14.

[11] D. M. Cooke, F. Alted et al. `https://github.com/pydata/numexpr`, viewed on the 4^{th} March 2018.

[12] `http://cython.org/`, viewed on the 20^{th} February 2018.

[13] G. Da Prato and J. Zabczyk. *Stochastic equations in infinite dimensions.* Cambridge University Press, 1992. [14] A. Erdélyi, W. Magnus, F. Oberhettinger, F. G. Tricomi. *Higher Transcendental Functions, vol. I.* McGraw-Hill, New York, 1953.

[15] J. Gatheral, T. Jaisson and M. Rosenbaum. *Volatility is rough.* Preprint, arXiv: 1410.3394, 2014.

[16] P. Glassermann. *Monte Carlo Methods in Financial Engineering.* Springer, 2004.

[17] `https://www.hdfgroup.org/solutions/hdf5/`, viewed on the 10^{th} March 2018.

[18] `https://hgomersall.github.io/pyFFTW/`, viewed on the 12^{th} January 2018.

[19] M. Hoffmann. *Stochastische Integration. Eine Einführung in die Finanzmathematik.* Springer Spektrum, Wiesbaden 2016.

[20] A. Jentzen, P. E. Kloeden and A. Neuenkirch. *Pathwise approximation of stochastic differential equations on domains: higher order convergence rates without global Lipschitz coefficients.* Numer. Math., 112:41–64, 2009.

[21] F. Johansson and others.*mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.0)*,`http://mpmath.org/`, September 2017, viewed on the 2^{nd} April 2018.

[22] M. Keller-Ressel. *Financial mathematics I.* Lecture notes, TU Berlin, 2013.

[23] M. Keller-Ressel. *Stochastische Analysis.* Lecture notes, TU Dresden, 2015.

[24] W. König. *Wahrscheinlichkeitstheorie I und II.* Lecture notes, TU Berlin, 2012.

[25] R. Kruse. *Strong and Weak Approximation of Semilinear Stochastic Evolution Equations.* Springer, 2014.

[26] V. Kuznetsov. *Special Functions and Their Symmetries. Postgradu-*

ate Course in Applied Analysis. Lecture notes, University of Leeds, 2003.

[27] M. Ledoux. *Isoperimetry and Gaussian analysis.* Lectures on probability theory and statistics (Saint-Flour, 1994), volume 1648 of Lecture Notes in Math., pages 165–294. Springer, Berlin, 1996.

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

[29] R. McCrickerd and M. S. Pakkanen. *Turbocharging Monte Carlo pricing for the rough Bergomi model.* Preprint, arXiv:1708.02563, 2018.

[30] R. McCrickerd. `https://github.com/ryanmccrickerd/rough_bergomi`, viewed on the 15^{th} February 2018.

[31] Y. Mishura. *Stochastic Calculus for Fractional Brownian Motion and Related Processes.* Lecture Notes in Mathematics, Springer, Berlin-Heidelberg, 2008.

[32] I.P. Natanson. *Theorie der Funktionen einer reellen Veränderlichen.* Verlag Harri Deutsch, Thun, 4. Auflage, Berlin, 1975.

[33] A. Neuenkirch and T. Shalaiko. *The Order Barrier for Strong Approximation of Rough Volatility Models.* preprint, arXiv: 1606.03854v1, 2016.

[34] D. Nualart. *Fractional Brownian motion: stochastic calculus and applications.* Proceedings of the International Congress of Mathematicians, Madrid, 2006. European Math. Soc. Publ. House, Zürich.

[35] `https://pandas.pydata.org/`, viewed on the 14^{th} April 2018.

[36] `https://www.pytables.org/`, viewed on the 6^{th} March 2018.

[37] A.N. Shiryaev. *Probability.* Second edition. Graduate Texts in Mathematics,95. Springer-Verlag, New York, 1996.

[38] V. M. Sithi and S. C. Lim. *On the spectra of Riemann-Liouville fractional Brownian motion.* Journal of Physics A: Mathematical and General. 28: 2995 - 3003, 1995.

[39] J. Michael Steele. *Stochastic Calculus and Financial Applications.* Springer,

2003. [39] H. Thum. *Numerical Simulations of the Weak Approximation Error for Parabolic Stochastic Partial Differential Equations.* Master thesis, ETH Zürich, August 2013.

[41] L. Viitasaar. *Integration in a Normal World: Fractional Brownian Motion and Beyond*. Aalto University publication series DOCTORAL DISSERTATIONS 14/2014, Helsinki, 2014.

[42] L. C. Young. *An inequality of the Hölder type, connected with Stieltjes integration*. Acta Math., 67:251–282, 1936.

[43] M. Zähle. *Integration with respect to fractal functions and stochastic calculus. Part I.* Probability Theory and Related Fields, 111(2): 333 - 372, Springer, 1998.