

# Bayesian inference of stochastic volatility models used by market participants

**Lorenzo Proserpio**

Supervisor: Marco Maggis

External supervisor: Martin Simon

A thesis presented for the  
Master's degree in Mathematics



Università degli Studi di Milano  
Dipartimento di Matematica  
19 July 2023

# Acknowledgement

# Index

<b>I</b>	<b>Dataset</b>	<b>6</b>
<b>II</b>	<b>Heston</b>	<b>8</b>
<b>1</b>	<b>Heston Model</b>	<b>8</b>
<b>2</b>	<b>The Valuation Equation</b>	<b>9</b>
2.1	The market price of volatility risk . . . . .	10
<b>3</b>	<b>The Characteristic Formula</b>	<b>11</b>
3.1	Derivation of the pseudo-probabilities . . . . .	11
3.2	Numerical integration of the complex logarithm . . . . .	13
3.3	Derivation of the characteristic function . . . . .	14
<b>4</b>	<b>Fourier Cosine Expansion for Vanilla Options</b>	<b>14</b>
4.1	Inverse Fourier integral via cosine expansion . . . . .	15
4.2	Pricing european options . . . . .	16
4.3	Coefficients $V_k$ for european put options . . . . .	16
4.4	Computation of $a$ and $b$ . . . . .	18
<b>5</b>	<b>Calibration</b>	<b>18</b>
5.1	Analytical gradient . . . . .	19
5.2	Levenberg-Marquardt . . . . .	20
5.3	Numerical results . . . . .	22
<b>6</b>	<b>Simulation</b>	<b>23</b>
6.1	Cumulative distribution function of $v_T v_t$ . . . . .	23
6.2	Explicit solution for the log-asset price . . . . .	24
6.3	Gamma approximation scheme . . . . .	25
6.4	Implementation of caches . . . . .	25
<b>7</b>	<b>At-the-money forward skew</b>	<b>27</b>
<b>III</b>	<b>Rough Heston</b>	<b>31</b>
<b>1</b>	<b>Stylized empirical facts</b>	<b>31</b>
<b>2</b>	<b>Building the model</b>	<b>31</b>
2.1	Hawkes Processes . . . . .	32
2.2	Encoding the 2 <sup>nd</sup> property . . . . .	32
2.3	Encoding the 3 <sup>rd</sup> property . . . . .	33

2.4	Encoding the 1 <sup>st</sup> property . . . . .	33
2.5	Encoding the 4 <sup>th</sup> property . . . . .	34
2.5.1	Wiener-Hopf equations . . . . .	35
2.6	From microstructure to macrostructure . . . . .	35
2.7	Mittag-Leffler functions . . . . .	36
2.8	Adjusting the initial volatility . . . . .	37
<b>3</b>	<b>Rough Heston model</b>	<b>38</b>
3.1	Inference of $\lambda\gamma(\cdot)$ from the forward variance curve . . . . .	39
<b>4</b>	<b>The Characteristic function</b>	<b>40</b>
4.1	The Characteristic function of an Hawkes process . . . . .	40
4.2	Intuition about the result . . . . .	42
4.3	Rational approximation of the solution . . . . .	45
4.3.1	Small times expansion . . . . .	45
4.3.2	Long times expansion . . . . .	48
4.3.3	Padé approximation . . . . .	49
<b>5</b>	<b>Pricing</b>	<b>51</b>
5.1	Fourier transform technique for Vanilla Options . . . . .	52
5.2	Numerical implementation . . . . .	54
<b>6</b>	<b>Calibration</b>	<b>54</b>
6.1	Numerical results . . . . .	55
<b>7</b>	<b>Simulation</b>	<b>56</b>
7.1	The HQE scheme . . . . .	57
7.1.1	The Andersen Quadratic Exponential scheme . . . . .	58
7.1.2	Hybrid simulation step . . . . .	58
<b>8</b>	<b>Is Heston's ATMF skew problem solved?</b>	<b>59</b>
<b>IV</b>	<b>Bayesian Inference</b>	<b>62</b>
<b>1</b>	<b>Additional models</b>	<b>62</b>
1.1	Rough Bergomi model . . . . .	62
1.1.1	Calibration and Numerical Results . . . . .	63
1.2	Quintic Ornstein Uhlenbeck model . . . . .	64
1.2.1	Calibration and Numerical Results . . . . .	64
<b>2</b>	<b>Approximate Bayesian Computation</b>	<b>65</b>
2.1	Sliced Wasserstein distance . . . . .	66

<b>3</b>	<b>Infer the models</b>	<b>67</b>
3.1	Numerical results . . . . .	67
3.2	Calibration using ABC . . . . .	69
3.2.1	ABC Heston calibration . . . . .	69

# Introduction

## Part I

# Dataset

This thesis wants to be somehow usable by practitioners. Everything that we will show from a theoretical point of view is also implemented and used on real market data. The data used are the SPX options chain of the 23<sup>rd</sup> January 2023 (source: Bloomberg), with moneyness from 80% to 120% and tenors from 2 weeks to almost 10 years. The closing price of that day was 4019.81.

## Implied drift term

Something that is usually assumed in academic papers is that the forward is flat. This means that we are implicitly assuming that there is no drift term in the risk-free world dynamics. This is reasonable in a low rates environment and if we are considering an underlying with a very low yield. As you can imagine, this is not acceptable for us, especially since the high rates environment of the last year, so we will take great care into keeping the drift term in all of our calculations. The drift term is the difference between the risk-free rate and the yield, continuously compounded, of the underlying. To be completely precise it includes not only the yield, but also the borrowing cost. In order to estimate that we recall the call-put parity. We will use the following notation:

- $C_{t,K}$  is the value of a call at time  $t$  with strike  $K$  and tenor  $T$ ;
- $P_{t,K}$  is the value of a put at time  $t$  with strike  $K$  and tenor  $T$ ;
- $S_t$  is the spot value of the underlying;
- $r$  is the risk-free rate (continuously compounded);
- $q$  is the yield of the underlying (continuously compounded).

Then the following hold:

$$C_{t,K} - P_{t,K} = S_t e^{-q(T-t)} - K e^{-r(T-t)}$$

solving for  $q$  we obtain:

$$q = \frac{1}{t-T} \log \left( \frac{C_{t,K} - P_{t,K} + K e^{-r(T-t)}}{S_t} \right)$$

now this relation still holds for  $t = 0$  and we can observe the market price of spot  $S_0$ , the call market-price  $\hat{C}_{0,K}$  and the put price  $\hat{P}_{0,K}$ . So the relation now is:

$$q = -\frac{1}{T} \log \left( \frac{\hat{C}_{0,K} - \hat{P}_{0,K} + K e^{-rT}}{S_0} \right)$$

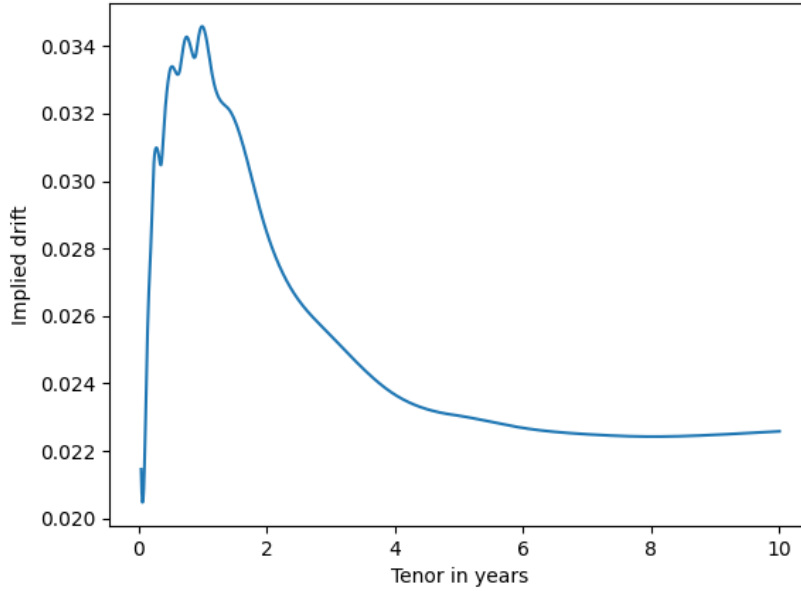
one can argue now that  $q$  can depend on the strike, but this is not true. Indeed, let's suppose that this is true and consider two strikes  $K_1 < K_2$  with  $q_{K_1} > q_{K_2}$ . Buy the following portfolio at time  $t = 0$ :

$$\frac{1}{S_0}(C_{0,K_1} - P_{0,K_1} + K_1 e^{-rT}) - \frac{1}{S_0}(C_{0,K_2} - P_{0,K_2} + K_2 e^{-rT})$$

due to call-put parity buying this portfolio at time zero let you receive a premium, indeed its value at time 0 is

$$\frac{(C_{0,K_1} - P_{0,K_1} + K_1 e^{-rT})}{S_0} - \frac{(C_{0,K_2} - P_{0,K_2} + K_2 e^{-rT})}{S_0} = e^{-q_{K_1}T} - e^{-q_{K_2}T} < 0$$

At maturity the value of the portfolio is 0, because  $C_{T,K_1} - P_{T,K_1} + K_1 = S_T = C_{T,K_2} - P_{T,K_2} + K_2$ . So this strategy is an arbitrage. The same, with obvious modifications, applies if  $q_{K_1} < q_{K_2}$ . So in conclusion the term  $q$  depends only on the tenor. To estimate the risk-free rate we used the values of the zero coupon USD OIS Swap rates, interpolated with cubic splines for the missing tenors. Here below there is the curve for the implied drift obtained, as we can see such curve is not zero at all.





## Part II

# Heston

The Heston model, named after its creator Steven Heston, was introduced for the first time in 1993 and has become widely used in the industry due to its ability to capture some important features of low-frequency asset price movements. The model is able to represent the following stylized effects observed in the market:

1. volatility tends to revert to some long-run level, which might itself be time-varying;
2. volatility tends to cluster, this refers to the bunching of large moves and small moves in the price process;
3. stock price returns tend to be negatively correlated with volatility;
4. implied volatility has a skew and a term structure.

We will show how to calibrate and simulate the Heston model in an efficient manner up to the standard of the industry. At the end of the chapter, we will also show its main limitation, which paved the way for developing more sophisticated models.

## 1 Heston Model

Let  $(\Omega, \{(F_t)_{t \geq 0}\}, \mathbb{P})$  a complete filtered probability space and let call  $\mathbb{P}$  the *physical-measure*. Let  $T < \infty$  be the right limit of our time horizon from now on. Given a stock price process  $S = (S_t)_{t \geq 0}$  the Heston model, under  $\mathbb{P}$ , is the following :

$$\begin{cases} dS_t = \mu S_t dt + S_t \sqrt{v_t} dW_t \\ dv_t = \kappa(\eta - v_t)dt + \theta \sqrt{v_t} d\tilde{W}_t \\ v_0 = \sigma_0^2 \end{cases}$$

where:

- $\mu$  is the drift of the stock returns;
- $W = (W_t)_{t \geq 0}$  and  $\tilde{W} = (\tilde{W}_t)_{t \geq 0}$  are two correlated Brownian motions with  $d\langle W, \tilde{W} \rangle_t = \rho dt$  and  $\rho \in [-1, 1]$ ;
- $\sigma_0 > 0$  is the initial volatility;
- $\eta > 0$  is the long run variance;
- $\kappa > 0$  is the mean reversion rate;
- $\theta > 0$  is the volatility of the volatility.

The variance process is strictly positive if  $2\kappa\eta > \theta^2$  (condition only sufficient, not necessary). This is known as *Feller condition*.

## 2 The Valuation Equation

Consider two independent claims which prices, at time  $t$ , are given as  $U_t = u(t, S_t, v_t)$  and  $\tilde{U}_t = \tilde{u}(t, S_t, v_t)$ . Suppose that  $u$  and  $\tilde{u}$  are  $\mathcal{C}^1(\mathbb{R}^+)$  w.r.t. the first variable and  $\mathcal{C}^2(\mathbb{R}^+ \times \mathbb{R}^+)$  w.r.t. the last two variables. Consider a portfolio consisting of a long contract  $U$ , short  $\Delta$  shares of the stock and short  $\Delta_1$  contracts of  $\tilde{U}$ . So the value (denoted with  $\Pi_t$ ) of our portfolio at time  $t$  is equal to:

$$\Pi_t = U_t - \Delta S_t - \Delta_1 \tilde{U}_t$$

then we can apply Itô's lemma and write the dynamics, omitting some subscripts, of the value of the portfolio as:

$$\begin{aligned} d\Pi_t = & \left\{ \frac{\partial U}{\partial t} + \frac{1}{2}vS^2 \frac{\partial^2 U}{\partial S^2} + \rho\theta vS \frac{\partial^2 U}{\partial S \partial v} + \frac{1}{2}\theta^2 v \frac{\partial^2 U}{\partial v^2} \right\} dt \\ & + \left\{ \frac{\partial \tilde{U}}{\partial t} + \frac{1}{2}vS^2 \frac{\partial^2 \tilde{U}}{\partial S^2} + \rho\theta vS \frac{\partial^2 \tilde{U}}{\partial S \partial v} + \frac{1}{2}\theta^2 v \frac{\partial^2 \tilde{U}}{\partial v^2} \right\} dt \\ & + \left\{ \frac{\partial U}{\partial S} - \Delta_1 \frac{\partial \tilde{U}}{\partial S} - \Delta \right\} dS \\ & + \left\{ \frac{\partial U}{\partial v} - \Delta_1 \frac{\partial \tilde{U}}{\partial v} \right\} dv \end{aligned}$$

to make our portfolio instantaneously risk-free we must impose the following equations to eliminate the  $dS$  and  $dv$  terms:

$$\begin{cases} \frac{\partial U}{\partial S} - \Delta_1 \frac{\partial \tilde{U}}{\partial S} - \Delta = 0 \\ \frac{\partial U}{\partial v} - \Delta_1 \frac{\partial \tilde{U}}{\partial v} = 0 \end{cases} \quad (1)$$

since our portfolio is now risk-free its value must be equal to the value of a portfolio with the same initial value invested in the cash market at risk-free rate  $r$ , so:

$$d\Pi_t = r\Pi_t dt = r(U_t - \Delta S_t - \Delta_1 \tilde{U}_t) dt$$

then choosing  $\Delta$  and  $\Delta_1$  as in (1) and collecting all the  $U$  terms on the LHS and all the  $\tilde{U}$  terms on the RHS we obtain (omitting once again some subscripts):

$$\begin{aligned} & \frac{\frac{\partial U}{\partial t} + \frac{1}{2}vS^2 \frac{\partial^2 U}{\partial S^2} + \rho\theta vS \frac{\partial^2 U}{\partial S \partial v} + \frac{1}{2}\theta^2 v \frac{\partial^2 U}{\partial v^2} + rS \frac{\partial U}{\partial S} - rU}{\frac{\partial U}{\partial v}} \\ & = \frac{\frac{\partial \tilde{U}}{\partial t} + \frac{1}{2}vS^2 \frac{\partial^2 \tilde{U}}{\partial S^2} + \rho\theta vS \frac{\partial^2 \tilde{U}}{\partial S \partial v} + \frac{1}{2}\theta^2 v \frac{\partial^2 \tilde{U}}{\partial v^2} + rS \frac{\partial \tilde{U}}{\partial S} - r\tilde{U}}{\frac{\partial \tilde{U}}{\partial v}} \end{aligned}$$

now since the LHS depends on  $U$  and the RHS depends on  $\tilde{U}$  then the only way for which this is possible is if they are equal to some function  $f = f(t, S_t, v_t)$  which is independent from  $U$  and  $\tilde{U}$ . So we obtain:

$$\frac{\partial U}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + \rho\theta vS\frac{\partial^2 U}{\partial S\partial v} + \frac{1}{2}\theta^2 v\frac{\partial^2 U}{\partial v^2} + rS\frac{\partial U}{\partial S} - rU = -f\frac{\partial U}{\partial v} \quad (2)$$

Equation (2) is called the valuation equation for  $U$ . WLOG we can rewrite  $f$  as:

$$f(t, S_t, v_t) = \kappa(\eta - v_t) - \sqrt{v_t}\phi(t, S_t, v_t)$$

where  $\phi_t$  is some arbitrary  $\mathbb{F}$ -adapted function and it is called the *market price of volatility risk*.

**NOTE:** the stock  $S$  is assumed to have yield equals to zero. If the stock in consideration has yield  $q$  we can simply switch  $r$  with  $(r - q)$  in the calculations before.

## 2.1 The market price of volatility risk

Without assuming that there are two traded independent claims with one of which is dependent from the variance process we cannot a priori fix an  $\mathbb{F}$ -adapted  $\phi$ . Indeed, we can rewrite  $\tilde{W} = \rho W + \sqrt{1 - \rho^2}W^\perp$  where  $W^\perp$  is a continuous Brownian motion on the filtration  $\mathcal{F}$  such that  $d\langle W, W^\perp \rangle = 0$ . Then we can rewrite the Heston model under  $\mathbb{P}$  as:

$$\begin{cases} dS_t = \mu S_t dt + S_t \sqrt{v_t} dW_t \\ dv_t = \kappa(\eta - v_t)dt + \theta\rho\sqrt{v_t}dW_t + \theta\sqrt{1 - \rho^2}\sqrt{v_t}dW_t^\perp \end{cases}$$

now take  $\lambda_t = \lambda(t, S_t, v_t)$  defined as:

$$\lambda_t := \frac{\mu - r}{\sqrt{v_t}}$$

and take  $\phi_t$   $\mathbb{F}$ -adapted such that the Novikov's condition is satisfied, i.e.;

$$\mathbb{E}\left[\exp\left(\frac{1}{2}\int_0^T[\lambda_t^2 + \phi_t^2]dt\right)\right] < \infty$$

then we have that the process  $Z = (Z_t)_{t \in [0, T]}$  defined as:

$$Z_t = \mathcal{E}(-\lambda_t W - \phi_t W^\perp)_t$$

where  $\mathcal{E}$  is the Doléans-Dade exponential is a  $\mathbb{P}$ -martingale, with  $E[Z_T] = 1$  and strictly positive. So, using the Radon-Nikodym's theorem we can define a measure  $\mathbb{Q}$  on  $\mathbb{F}$  as:

$$\left.\frac{d\mathbb{Q}}{d\mathbb{P}}\right|_{\mathcal{F}_t} := Z_t$$

and we have that  $\mathbb{Q}$  is equivalent to  $\mathbb{P}$  on  $F_T$  and moreover is martingale equivalent measure. By Girsanov's theorem we can define a 2-dimensional  $\mathbb{Q}$ -Brownian motion  $(W^{\mathbb{Q}}, W^{\mathbb{Q},\perp})$  as:

$$W_t^{\mathbb{Q}} = W_t + \int_0^t \lambda_s ds \quad \text{and} \quad W_t^{\mathbb{Q},\perp} = W_t + \int_0^t \phi_s ds$$

and under such a  $\mathbb{Q}$  we have that the Heston model has the following dynamics:

$$\begin{cases} dS_t = rS_t dt + S_t \sqrt{v_t} dW_t^{\mathbb{Q}} \\ dv_t = [\kappa(\eta - v_t) - \theta \sqrt{v_t}(\rho \lambda_t + \sqrt{1 - \rho^2} \phi_t)] dt + \theta \sqrt{v_t} d\tilde{W}_t^{\mathbb{Q}} \\ v_0 = \sigma_0^2 \end{cases}$$

and under this we have that the discounted stock price is a  $\mathbb{Q}$  local martingale. However, we can choose any arbitrary  $\phi$  and obtain another  $\hat{\mathbb{Q}}$  with the same properties. But, as in our case, both the claims are actively traded we can infer the  $\phi$  from the market prices of the options, which will determine a  $\mathbb{Q}$  and then fix the  $\phi$ . Among all the possible choices from now on we will choose  $\phi = 0$ . The resulting  $\mathbb{Q}^M$  is called the *minimal martingale measure* and Föllmer and Schweizer have proven that, for models with continuous price trajectories, solves the minimization problem:

$$\mathbb{Q}^M = \arg \min_{\mathbb{Q} \in \mathcal{M}} \mathbb{H}(\mathbb{Q}|\mathbb{P})$$

where  $\mathcal{M}$  is the set of equivalent martingale measures and  $\mathbb{H}$  is the reverse relative entropy. Indeed this is exactly what we obtain if we can completely infer the  $\phi$  from the market prices of the options (because we think that  $\mathbb{P}$  contains all the information to reconstruct  $\mathbb{Q}$ ).

**NOTE:** From now on, to lighten the notation, our Heston model under  $\mathbb{Q}$  is written in the following way:

$$\begin{cases} dS_t = (r - q)S_t dt + S_t \sqrt{v_t} dW_t \\ dv_t = \kappa(\eta - v_t) dt + \theta \sqrt{v_t} d\tilde{W}_t \\ v_0 = \sigma_0^2 \end{cases}$$

with  $W = (W_t)_{t \geq 0}$  and  $\tilde{W} = (\tilde{W}_t)_{t \geq 0}$  are two correlated  $\mathbb{Q}$  Brownian motions with  $d\langle W, \tilde{W} \rangle_t = \rho dt$ .

### 3 The Characteristic Formula

#### 3.1 Derivation of the pseudo-probabilities

Setting the  $\phi = 0$  we have that the valuation equation for our model, to price a call option  $C$ , becomes:

$$\frac{\partial C}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 C}{\partial S^2} + \rho \theta v S \frac{\partial^2 C}{\partial S \partial v} + \frac{1}{2} \theta^2 v \frac{\partial^2 C}{\partial v^2} + r S \frac{\partial C}{\partial S} - r C + [\kappa(\eta - v)] \frac{\partial C}{\partial v} = 0$$

if we now substitute with  $\tau = T - t$  and  $x = \log\left(\frac{S_t e^{(\tau-q)\tau}}{K}\right)$  in the previous equation we obtain:

$$-\frac{\partial C}{\partial \tau} + \frac{1}{2}v \frac{\partial^2 C}{\partial x^2} + \rho\theta v \frac{\partial^2 C}{\partial x \partial v} + \frac{1}{2}\theta^2 v \frac{\partial^2 C}{\partial v^2} - \frac{1}{2}v \frac{\partial C}{\partial x} + [\kappa(\eta - v)] \frac{\partial C}{\partial v} = 0 \quad (3)$$

according to Duffie, Pan and Singleton (2000), the solution to this equation has the form of:

$$C(x, v, \tau) = K \{e^x P_1(x, v, \tau) - P_0(x, v, \tau)\} \quad (4)$$

notice how this remind us to the solution of the Black&Scholes' Equation. Moreover, the  $P_j$  are absolutely continuous, differentiable and both  $P_j$  and  $P'_j$  are absolutely integrable on  $\mathbb{R}$ . Putting (4) into (3) we obtain that  $P_j$  with  $j = 0, 1$  must satisfy the following PDE:

$$-\frac{\partial P_j}{\partial \tau} + \frac{1}{2}v \frac{\partial^2 P_j}{\partial x^2} - \left(\frac{1}{2} - j\right)v \frac{\partial P_j}{\partial x} + \frac{1}{2}\theta^2 v \frac{\partial^2 P_j}{\partial v^2} + \rho\theta v \frac{\partial^2 P_j}{\partial x \partial v} + [\kappa(\eta+1) - j\rho\theta v] \frac{\partial P_j}{\partial v} = 0$$

with boundary conditions:

$$\lim_{\tau \rightarrow 0} P_j(x, v, \tau) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Now take the Fourier transform of  $P_j$  as:

$$\hat{P}_j(u, v, \tau) = \int_{\mathbb{R}} e^{-iux} P_j(x, v, \tau) dx \quad \text{then} \quad \hat{P}_j(u, v, 0) = \frac{1}{iu}$$

and the inverse transform given by:

$$P_j(x, v, \tau) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-iux} \hat{P}_j(u, v, \tau) du$$

thanks to the regularity of  $P_j$  we have that  $[\widehat{P_j(u)'}] = iu\hat{P}_j(u)$ . So, using this property, we can rewrite our equation as:

$$-\left[\frac{1}{2}u^2 - \left(\frac{1}{2} - j\right)iu\right]v\hat{P}_j + \left\{\rho\theta iuv + [\kappa(\eta+1) - j\rho\theta v]\right\} \frac{\partial \hat{P}_j}{\partial v} + \frac{1}{2}\theta^2 v \frac{\partial^2 \hat{P}_j}{\partial v^2} = \frac{\partial \hat{P}_j}{\partial \tau}$$

now take two arbitrary functions  $f(u, \tau)$ ,  $g(u, \tau)$ . The solution should be of the form:

$$\hat{P}_j(u, v, \tau) = \frac{1}{iu} \exp\{f(u, \tau)\eta + g(u, \tau)v\}$$

it follows that:

$$\begin{aligned} \frac{\partial \hat{P}_j}{\partial \tau} &= \left\{ \eta \frac{\partial f}{\partial \tau} + v \frac{\partial g}{\partial \tau} \right\} \hat{P}_j \\ \frac{\partial \hat{P}_j}{\partial v} &= g \hat{P}_j \\ \frac{\partial^2 \hat{P}_j}{\partial v^2} &= g^2 \hat{P}_j \end{aligned}$$

now define:

$$\begin{aligned}\alpha &= -\frac{u^2}{2} - \frac{i u}{2} + i j u \\ \beta &= \kappa - \rho \theta (j + i u) \\ \gamma &= \frac{\theta^2}{2}\end{aligned}$$

then it must be:

$$\begin{aligned}\frac{\partial f}{\partial \tau} &= \kappa g \\ \frac{\partial g}{\partial \tau} &= \alpha - \beta g - \gamma g^2 \\ &= \gamma (g - r_+)(g - r_-)\end{aligned}$$

where  $r_{\pm} = \frac{\beta \pm \sqrt{\beta^2 - 4\alpha\gamma}}{2\gamma} =: \frac{\beta \pm d}{\theta^2}$ . Integrating and putting the terminal condition  $f(u, 0) = 0$  and  $g(u, 0) = 0$ , we obtain:

$$\begin{aligned}f(u, \tau) &= \kappa \left\{ r_- \tau - \frac{2}{\theta^2} \log \left( \frac{1 - \frac{r_-}{r_+} e^{-d\tau}}{1 - \frac{r_-}{r_+}} \right) \right\} \\ g(u, \tau) &= r_- \frac{1 - e^{-d\tau}}{1 - \frac{r_-}{r_+} e^{-d\tau}}\end{aligned} \tag{5}$$

Taking the inverse transform on  $\hat{P}_j$  we obtain finally the form of the pseudo probabilities as:

$$P_j(x, v, \tau) = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \Re \left\{ \frac{\exp \{ f_j(u, \tau) \eta + g_j(u, \tau) v + i u x \}}{i u} \right\} du \tag{6}$$

### 3.2 Numerical integration of the complex logarithm

In (5) we decided to take  $r_-$  to define  $f$ , but we could have taken  $r_+$  and everything would have been "almost" the same (this definition coincides with our previous one only if the imaginary part of the complex logarithm is chosen so that  $f(u, \tau)$  is continuous with respect to  $u$ ). However, we are interested in integrating numerically the characteristic function of the Heston model. We have to keep in mind that the complex logarithm has a branch (we decided to take as branch the semiaxis of negative real numbers in the complex plane), so in order to avoid any kind of discontinuity we want that the characteristic function never crosses the negative real axis on  $(0, \infty)$ . Albrecher, Mayer, Schoutens, Tistaert (2006) proved the following result when using the FFT-like approach:

**Proposition 1.** *Whenever the parameters of the Heston model are such that  $\Im\{d(u)\} := \Im\{\sqrt{(\rho\theta ui - \kappa)^2 + \theta^2(iu + u^2)}\} \neq 0$  and  $2\kappa\eta \neq \theta^2 n$  (where  $n \in \mathbb{N}$ ), then defining (5) using  $r_+$  leads to numerical instabilities for sufficiently large maturities.*

In order to use methods which leverage the frequency domain we usually have to evaluate the characteristic function in  $u - (\alpha + 1)i$  for positive  $u$ . So they also proved the following proposition:

**Proposition 2.** *Denote with  $\phi(u)$  the characteristic function of the Heston model obtained using  $r_-$  in (5). Then  $\forall \alpha > 0$  and  $\forall u \in (0, \infty)$  the function  $\phi(u - (\alpha + 1)i)$  does not cross the negative real axis.*

In conclusion if we use  $r_-$  in (5) then the characteristic function that we will obtain in the next paragraph is suitable for numerical integration.

### 3.3 Derivation of the characteristic function

By definition the characteristic function under  $\mathbb{Q}$  is defined as:

$$\phi(u) = \mathbb{E}_{\mathbb{Q}} \left[ e^{iux\tau} \middle| x_t = \log \left( \frac{S_t e^{(r-q)\tau}}{K} \right); v_t \right]$$

now thanks to (6) we know that:

$$\mathbf{Pr}_{\mathbb{Q}}(x_T > x_t) = P_0(x_t, v_t, \tau)$$

so, if we define  $k = -x_t$  the density is:

$$\begin{aligned} p(k) &= -\frac{\partial P_0}{\partial k} \\ &= \frac{1}{2\pi} \int_{\mathbb{R}} \exp \{f_0(s, \tau)\eta + g_0(s, \tau)v_t - isk\} ds \end{aligned}$$

then we have that, using Fubini theorem:

$$\begin{aligned} \phi_H(u; \tau, x_t, v_t) &= \int_{\mathbb{R}} e^{iuk + iux_t} p(k) dk \\ &= \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} \exp \{f_0(s, \tau)\eta + g_0(s, \tau)v_t\} e^{iuk} e^{-isk} e^{iux_t} ds dk \\ &= \frac{e^{iux_t}}{2\pi} \int_{\mathbb{R}} \exp \{f_0(s, \tau)\eta + g_0(s, \tau)v_t\} \int_{\mathbb{R}} e^{ik(u-s)} dk ds \\ &= e^{iux_t} \int_{\mathbb{R}} \exp \{f_0(s, \tau)\eta + g_0(s, \tau)v_t\} \delta(u-s) ds \\ &= \exp \{f_0(u, \tau)\eta + g_0(u, \tau)v_t + iu \log [S_t e^{(r-q)\tau}] - iu \log(K)\} \end{aligned}$$

## 4 Fourier Cosine Expansion for Vanilla Options

In order to calibrate our model we need a way to price vanilla options (calls & puts) which usually require integrating the probability density function. However, since we have its Fourier transform (the characteristic function) we can

leverage the *Fourier Cosine Expansion* method developed by F. Fang & C.W. Osterlee. The computational speed, especially for plain vanilla options, makes this integration method state-of-the-art for calibration at financial institutions. From now on, in this section,  $t_0 = 0$ , the final time is  $T$ , we will call the Heston transition density function  $f_H$  and  $K$  will be always used for the strike.

#### 4.1 Inverse Fourier integral via cosine expansion

Remember that:

$$\phi_H(u) = \int_{\mathbb{R}} e^{iux} f_H(x) dx$$

we are interested in approximating:

$$f_H(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-iux} \phi_H(u) du$$

Since  $f_H \in \mathcal{L}^1(\mathbb{R})$  and it is  $\geq 0$  we are able to choose a finite interval  $[a, b]$  such that we have:

$$\tilde{\phi}_H(u) = \int_{\mathbb{R}} e^{iux} f_H(x) \mathbb{1}_{[a,b]}(x) dx = \int_a^b e^{iux} f_H(x) dx \approx \int_{\mathbb{R}} e^{iux} f_H(x) dx = \phi_H(u)$$

Now the function  $\tilde{f}_H := f_H \mathbb{1}_{[a,b]}$  is obviously supported on a finite interval  $[a, b]$  and it is continuously differentiable on that interval (look at how it is defined  $\phi_H$  and remember that we choose the root in order to have that continuous), so it admits a cosine expansion:

$$\tilde{f}_H(x) = \frac{\tilde{A}_0}{2} + \sum_{k=1}^{\infty} \tilde{A}_k \cos\left(k\pi \frac{x-a}{b-a}\right)$$

where:

$$\tilde{A}_k = \frac{2}{b-a} \int_a^b f_H(s) \cos\left(k\pi \frac{s-a}{b-a}\right) ds$$

comparing the two equations before we obtain:

$$\tilde{A}_k = \frac{2}{b-a} \Re\left\{\tilde{\phi}_H\left(\frac{k\pi}{b-a}\right) \cdot e^{-\frac{ik\pi a}{b-a}}\right\} \approx \frac{2}{b-a} \Re\left\{\phi_H\left(\frac{k\pi}{b-a}\right) \cdot e^{-\frac{ik\pi a}{b-a}}\right\} =: A_k$$

so if we replace  $\tilde{A}_k$  with  $A_k$  we have that:

$$\tilde{f}_H(x) \approx \frac{A_0}{2} + \sum_{k=1}^{\infty} A_k \cos\left(k\pi \frac{x-a}{b-a}\right) \approx f_H(x)$$

now since  $f_H$  is an *entire function* (function without any singularities anywhere in the complex plane, except at  $\infty$ ) the cosine Fourier expansion converges with exponential speed then we can choose a suitable  $N \in \mathbb{N}$  such that:

$$\frac{A_0}{2} + \sum_{k=1}^{\infty} A_k \cos\left(k\pi \frac{x-a}{b-a}\right) \approx \frac{A_0}{2} + \sum_{k=1}^{N-1} A_k \cos\left(k\pi \frac{x-a}{b-a}\right)$$



so in conclusion we obtain:

$$f_H(x) \approx \frac{A_0}{2} + \sum_{k=1}^{N-1} A_k \cos\left(k\pi \frac{x-a}{b-a}\right)$$

## 4.2 Pricing european options

Let  $x := \log\left(\frac{S_0}{K}\right)$  and  $y := \log\left(\frac{S_T}{K}\right)$  then consider a european option and call its value at expiration as  $v(y, T)$ . Given the transition density  $f_H(y|x)$  we have that the value of the claim at time 0 should be:

$$v(x, 0) = e^{-rT} \int_{\mathbb{R}} v(y, T) f(y|x) dy$$

Since the density rapidly decays to zero as  $y \rightarrow \pm\infty$  we truncate the infinite integration range without losing significant accuracy to  $[a, b]$  and we obtain approximation:

$$v(x, 0) \approx e^{-rT} \int_a^b v(y, T) f(y|x) dy$$

now we replace  $f_H(y|x)$  with the approximation obtained in the previous paragraph:

$$v(x, 0) \approx e^{-rT} \int_a^b v(y, T) \left[ \frac{A_0(x)}{2} + \sum_{k=1}^{N-1} A_k(x) \cos\left(k\pi \frac{y-a}{b-a}\right) \right] dy$$

now we define:

$$V_k := \frac{2}{b-a} \int_a^b v(y, T) \cos\left(k\pi \frac{y-a}{b-a}\right) dy$$

and obtain:

$$v(x, 0) \approx \frac{b-a}{2} e^{-rT} \left[ \frac{A_0(x) V_0}{2} + \sum_{k=1}^{N-1} A_k(x) V_k \right]$$

substituting the  $A_k(x)$  with the formulation in the previous paragraph we finally obtain:

$$v(x, 0) \approx e^{-rT} \left[ \frac{1}{2} \Re\{\phi_H(0; x)\} V_0 + \sum_{k=1}^{N-1} \Re\left\{ \phi_H\left(\frac{k\pi}{b-a}; x\right) \cdot e^{-\frac{ik\pi a}{b-a}} \right\} V_k \right]$$

## 4.3 Coefficients $V_k$ for european put options

The value of a put option at maturity, keeping the notation of the previous paragraph, is:

$$v(y, T) = [K(1 - e^y)]^+$$

so we obtain that  $V_k$  for a put can be expressed as:

$$V_k = \frac{2K}{b-a} \int_a^0 (1 - e^y) \cos \left( k\pi \frac{y-a}{b-a} \right) dy$$

this is the formulation given by Fang & Osterlee. However, Le Floc'h underlined that the  $V_k$  are computed relatively to the strike price, but the truncation range is relative to the spot price. The consequence is that very deep out-the-money & in-the-money puts are severely mispriced, especially for short maturities. Le Floc'h proposed an improved version to calculate the  $V_k$ , we will present here a reformulation of that method. We will compute the  $V_k$  this time with respect to the spot price. Let's define  $z := \log \left( \frac{S_T}{S_0} \right)$ , then we have that:

$$v(z, T) = S_0 \left[ \frac{K}{S_0} - e^z \right]^+ = S_0 [e^{-x} - e^z]^+$$

so we have that (notice that it is possible to price puts with  $-x \in (a, b)$ ):

$$\begin{aligned} V_k &= \frac{2S_0}{b-a} \int_a^b [e^{-x} - e^z]^+ \cos \left( k\pi \frac{z-a}{b-a} \right) dz \\ &= \frac{2S_0}{b-a} \int_a^{-x} [e^{-x} - e^z] \cos \left( k\pi \frac{z-a}{b-a} \right) dz \\ &= \frac{2S_0 e^{-x}}{b-a} \int_a^{-x} [1 - e^{z+x}] \cos \left( k\pi \frac{z-a}{b-a} \right) dz \\ &= \frac{2S_0 e^{-x}}{b-a} \int_a^{-x} [1 - e^{z+x}] \cos \left( k\pi \frac{z-a}{b-a} \right) dz \\ &= \frac{2}{b-a} [K\psi_k(a, -x) - S_0\chi_k(a, -x)] \end{aligned}$$

where:

$$\begin{aligned} \psi_k(a, -x) &:= \int_a^{-x} \cos \left( k\pi \frac{z-a}{b-a} \right) dz \\ \chi_k(a, -x) &:= \int_a^{-x} e^z \cos \left( k\pi \frac{z-a}{b-a} \right) dz \end{aligned}$$

after integration we obtain:

$$\begin{aligned} \psi_k(a, -x) &= \begin{cases} \frac{a-b}{k\pi} \sin \left( k\pi \frac{x+a}{b-a} \right) & k \neq 0 \\ -x-a & k = 0 \end{cases} \\ \chi_k(a, -x) &= \frac{1}{1 + \left( \frac{k\pi}{b-a} \right)^2} \left[ e^{-x} \cos \left( k\pi \frac{x+a}{b-a} \right) - e^a - \frac{k\pi}{b-a} e^{-x} \sin \left( k\pi \frac{x+a}{b-a} \right) \right] \end{aligned}$$

Now the only thing which is missing is how to compute the extremes of the truncation interval.

#### 4.4 Computation of $a$ and $b$

We used the formula proposed by Fang & Osterlee to compute the truncation range as:

$$[a, b] = [c_1 - 12\sqrt{|c_2|}, c_1 + 12\sqrt{|c_2|}]$$

where  $c_1$  and  $c_2$  are the two first cumulants. Indeed, defining the cumulant generating function as

$$g(u) := \log \phi_H(-iu)$$

we have that  $c_1 = g'(0)$  and  $c_2 = g''(0)$ . However, the second cumulant is easier to compute numerically than analytically. Here below there are the formulae, the first one is exact, the second one is obtained through Taylor expansion of the characteristic function:

$$c_1 = (r - q)\tau - \frac{\sigma_0}{2}$$

$$\begin{aligned} c_2 = & \frac{\sigma_0}{4\kappa^3} \{4\kappa^2[1 + e^{-\kappa\tau}(\rho\theta\tau - 1)] + \kappa[4\rho\theta(e^{-\kappa\tau} - 1) - 2\theta^2\tau e^{-\kappa\tau}] + \theta^2(1 - e^{-2\kappa\tau})\} \\ & + \frac{\eta}{8\kappa^3} \{8\kappa^3\tau - 8\kappa^2[1 + \rho\theta\tau + e^{-\kappa\tau}(\rho\theta\tau - 1)] \\ & + 2\kappa[(1 + 2e^{-\kappa\tau})\theta^2\tau + 8\rho\theta(1 - e^{-\kappa\tau})] + \theta^2(e^{-2\kappa\tau} + 4e^{-\kappa\tau} - 5)\} \end{aligned}$$

### 5 Calibration

In this section we present the Yiran Cui's method for a fast calibration of the Heston model. The calibration problem is simply to choose a set of parameters  $\Xi := [\sigma_0^2, \eta, \rho, \kappa, \theta]^\top$  such that the difference between the observed price of the calls and the puts and the price given by the Heston model with that parameters is minimum. We denote with  $C^*(K_i, T_i)$  the market prices for calls with strike  $K_i$  and maturity  $T_i$  and with  $C(\Xi; K_i, T_i)$  the prices for calls under Heston model with parameters  $\Xi$ . From now on we will focus only on calls, but the discussion is analogue on puts (indeed we will derive the gradient w.r.t. the parameters for calls, but using the put-call parity is the same for puts). Given  $n$  call options we define:

$$r_i(\Xi) := C(\Xi; K_i, T_i) - C^*(K_i, T_i) \quad i = 1, \dots, n$$

and the residual vector  $r(\Xi) = [r_1(\Xi), \dots, r_n(\Xi)]^\top$ . With this notation the calibration of the Heston model is an inverse problem in the nonlinear least square form as:

$$(\star) \quad \min_{\Xi} \frac{1}{2} \|r(\Xi)\|^2$$

since we suppose to have  $n \gg 5$  (where 5 is the number of parameters that we have to determine) it is an overdetermined problem. To tackle this kind of problem we will use the Levenberg-Marquardt method, which requires to

find the analytical gradient of the call price w.r.t. the parameters. Notice also that, in reality, we will use as the residual vector of the difference between the model implied volatility and the market observed implied volatility, but it actually makes no difference from a theoretical perspective since it is a one to one mapping with the prices of the calls.

## 5.1 Analytical gradient

In order to obtain the analytical gradient Cui uses an equivalent form for the Heston characteristic function which is easier to derive and it is still numerically continuous. First of all define:

$$\begin{aligned}\xi &:= \kappa - \theta \rho i u \\ d &:= \sqrt{\xi^2 + \theta^2(u^2 + iu)} \\ A &:= \frac{A_1}{A_2} = \frac{(u^2 + iu) \sinh \frac{Td}{2}}{d \cosh \frac{Td}{2} + \xi \sinh \frac{Td}{2}} \\ D &:= \log d + \frac{(k-d)T}{2} - \log \left( \frac{d+\xi}{2} + \frac{d-\xi}{2} e^{-dT} \right) =: \log B\end{aligned}$$

then the new form of the  $\phi_H$  is:

$$\begin{aligned}\phi_H(\Xi; u, T) &= \exp \left\{ iu[\log S_0 + (r-q)T] - \frac{T\kappa\eta\rho iu}{\theta} \right\} \\ &\times \exp \left\{ -\sigma_0^2 A + \frac{2\kappa\eta}{\theta^2} D \right\}\end{aligned}$$

Since it is only an algebrical manipulation from the previous characteristic function we will omit how it is obtained. Deriving we obtain that  $\nabla \phi_H(\Xi; u, T) = \phi_H(\Xi; u, T) \mathbf{h}(u)$  where  $\mathbf{h}(u) := [h_1(u), \dots, h_5(u)]^\top$  with elements:

$$\begin{aligned}h_1(u) &= -A \\ h_2(u) &= \frac{2\kappa}{\theta^2} D - \frac{T\kappa\rho iu}{\theta} \\ h_3(u) &= -\sigma_0^2 \frac{\partial A}{\partial \rho} + \frac{2\kappa\eta}{\theta^2 d} \left( \frac{\partial d}{\partial \rho} - \frac{d}{A_2} \frac{\partial A_2}{\partial \rho} \right) - \frac{T\kappa\eta iu}{\theta} \\ h_4(u) &= \frac{\sigma_0^2}{\theta iu} \frac{\partial A}{\partial \rho} + \frac{2\eta}{\theta^2} D + \frac{2\kappa\eta}{\theta^2 B} \frac{\partial B}{\partial \kappa} - \frac{T\eta\rho iu}{\theta} \\ h_5(u) &= -\sigma_0^2 \frac{\partial A}{\partial \theta} - \frac{4\kappa\eta}{\theta^3} D + \frac{2\kappa\eta}{\theta^2 d} \left( \frac{\partial d}{\partial \theta} - \frac{d}{A_2} \frac{\partial A_2}{\partial \theta} \right) + \frac{T\kappa\eta\rho iu}{\theta^2}\end{aligned}$$

computing also the partial derivatives in the previous formulae we have that:

$$\begin{aligned}
\frac{\partial d}{\partial \rho} &= -\frac{\xi \theta i u}{d} \\
\frac{\partial A_2}{\partial \rho} &= -\frac{\theta i u(2 + T\xi)}{2d} \left( \xi \cosh \frac{Td}{2} + d \sinh \frac{Td}{2} \right) \\
\frac{\partial A_1}{\partial \rho} &= -\frac{i u(u^2 + i u) T \xi \theta}{2d} \cosh \frac{Td}{2} \\
\frac{\partial A}{\partial \rho} &= \frac{1}{A_2} \frac{\partial A_1}{\partial \rho} - \frac{A}{A_2} \frac{\partial A_2}{\partial \rho} \\
\frac{\partial B}{\partial \kappa} &= \frac{i e^{\kappa T/2}}{\theta u} \left( \frac{1}{A_2} \frac{\partial d}{\partial \rho} - \frac{d}{A_2^2} \frac{\partial A_2}{\partial \rho} \right) + \frac{TB}{2}
\end{aligned}$$

and the last ones:

$$\begin{aligned}
\frac{\partial d}{\partial \theta} &= \left( \frac{\rho}{\theta} - \frac{1}{\xi} \right) \frac{\partial d}{\partial \rho} + \frac{\theta u^2}{d} \\
\frac{\partial A_1}{\partial \theta} &= \frac{(u^2 + i u) T}{2} \frac{\partial d}{\partial \theta} \cosh \frac{Td}{2} \\
\frac{\partial A_2}{\partial \theta} &= \frac{\rho}{\theta} \frac{\partial A_2}{\partial \rho} - \frac{2 + T\xi}{i u T \xi} \frac{\partial A_1}{\partial \rho} + \frac{\theta T A_1}{2} \\
\frac{\partial A}{\partial \theta} &= \frac{1}{A_2} \frac{\partial A_1}{\partial \theta} - \frac{A}{A_2} \frac{\partial A_2}{\partial \theta}
\end{aligned}$$

so now we can use the formula with the pseudo-probabilities calculated in the section 3.1 to obtain the gradient for a european call option at maturity and, as previously said, for a european put option, w.r.t. the parameters  $\Xi$ :

$$\begin{aligned}
\nabla C(\Xi; K, T) &= \frac{e^{-rT}}{\pi} \left[ \int_0^\infty \Re \left\{ \frac{K^{-iu}}{iu} \nabla \phi_H(\Xi; u - i, T) \right\} du \right. \\
&\quad \left. - K \int_0^\infty \Re \left\{ \frac{K^{-iu}}{iu} \nabla \phi_H(\Xi; u, T) \right\} du \right] \quad (7)
\end{aligned}$$

in order to evaluate the two integrals in (7) in the practical implementation we will use the Gauss-Legendre quadrature with 60 nodes and leveraging the fact that the integrands decay fast enough to justify a truncation of the integral domain to  $[0, 100]$ .

## 5.2 Levenberg-Marquardt

Let  $J = \nabla r^\top \in \mathbb{R}^{5 \times n}$  be the Jacobian matrix of the residual vector then for how it is defined the residual vector we have that:

$$J = [J_{ji}]_{j=1, \dots, 5}^{i=1, \dots, n} = \left[ \frac{\partial C(\Xi; K_i, T_i)}{\partial \Xi_j} \right]_{j=1, \dots, 5}^{i=1, \dots, n}$$

now let  $H(r_i) = \nabla \nabla^\top r_i \in \mathbb{R}^{5 \times 5}$  be the Hessian matrix of each residual  $r_i$ . Then the gradient and Hessian of the objective function  $f$  of the problem  $(\star)$  are:

$$\begin{aligned}\nabla f &= Jr \\ \nabla \nabla^\top f &= JJ^\top + \sum_{i=1}^n r_i H(r_i)\end{aligned}$$

The Levenberg-Marquardt algorithm is an iterative algorithm suitable for solving nonlinear least square problems. It is an hybrid between the Gauss-Newton algorithm and the steepest descent, however it is more robust than the first one. The stopping criterion for the LM algorithm is when one of the following is satisfied:

1.  $\|r(\Xi_k)\| \leq \varepsilon_1$ , where  $\varepsilon_1 \in \mathbb{R}^+$ , so if the solution found is sufficiently near the real solution;
2.  $\|J_k\|_\infty \leq \varepsilon_2$ , where  $\varepsilon_2 \in \mathbb{R}^+$ , so if the gradient is sufficiently small;
3.  $\frac{\|\Xi_k - \Xi_{k-1}\|}{\|\Xi_k\|} \leq \varepsilon_3$ , where  $\varepsilon_3 \in \mathbb{R}^+$ , so if the update in parameters is too small.

In practice works as follow:

---

**Algorithm 1** Levenberg - Marquardt algorithm

---

- 1: Given the initial guess  $\Xi_0$ , compute  $\|r(\Xi_0)\|$  and  $J_0$ .
  - 2: Choose the initial damping factor as  $\mu_0 = \tau \max\{\text{diag}(J_0)\}$  and  $\nu_0 = 2$ .
  - 3: Set  $k = 0$ .
  - 4: **while** TRUE **do**
  - 5:   Compute  $\Delta \Xi_k = (J_k J_k^\top + \mu_k I)^{-1} J_k r(\Xi_k)$ .
  - 6:   Compute  $\Xi_{k+1} = \Xi_k + \Delta \Xi_k$  and  $\|r(\Xi_{k+1})\|$ .
  - 7:   Compute  $\delta_L = \Delta \Xi_k^\top [\mu \Delta \Xi_k + J_k r(\Xi_k)]$  and  $\delta_F = \|r(\Xi_k)\| - \|r(\Xi_{k+1})\|$ .
  - 8:   **if**  $\delta_L > 0$  and  $\delta_F > 0$  **then**
  - 9:     Accept the step: compute  $J_{k+1}, \mu_{k+1} = \mu_k, \nu_{k+1} = \nu_k$ .
  - 10:   **else**
  - 11:     Recalculate the step: set  $\mu_k = \mu_k \nu_k, \nu_k = 2\nu_k$  and go to 4.
  - 12:   **end if**
  - 13:   **if** At least one stopping criterion is met **then**
  - 14:     Break.
  - 15:   **end if**
  - 16:    $k = k + 1$ .
  - 17: **end while**
- 

As we can see when the iteration is far from the optimum we give a large value to  $\mu_k$ , which is called the *damping factor* and so the Hessian of the objective function is dominated by the scaled identity matrix, meanwhile when the iteration

is closed to the optimum the Hessian matrix is dominated by the Gauss-Newton approximation. In particular we say that:

$$\nabla \nabla^\top f \approx J J^\top$$

so we use the conjecture that near the optimum the problem is a *small residual problem* (in the sense that  $\sum_{i=1}^n r_i H(r_i)$  is negligible due to having small  $r_i$ ). This is sensible since we think that the Heston model is a good model to explain the smile and the skew of the volatility surface, so in a couple of words it is an appropriate model for the volatility. It can also be shown that solving the equation at step 5. of the algorithm is equivalent in finding the solution to the minimization problem:

$$\Xi_{k+1} := \arg \min_{z \in \mathbb{R}^5} \{ \|J_k^\top(z - \Xi_k) + r(\Xi_k)\|^2 + \mu_k \|z - \Xi_k\|^2 \}$$

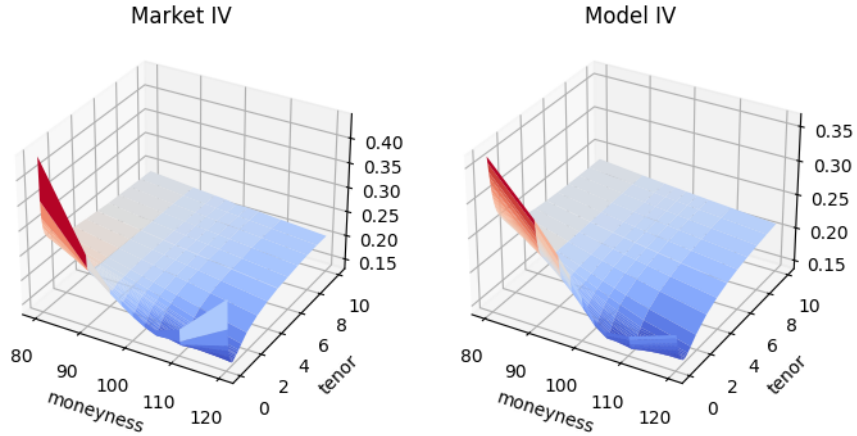
so whenever the  $\mu_k$  is sufficiently small we treat the problem as quasi-linear. Notice also that the constraints on parameters such as  $\rho \in [-1, 1]$ ,  $\sigma_0 > 0$ ,  $\kappa > 0$ ,  $\eta > 0$  and  $\theta > 0$  are not necessarily satisfied; indeed, whenever we have obtained the output  $\Xi$  we need to do a sanity check.

### 5.3 Numerical results

As a metric to compare the goodness of the fit we choose the mean relative percentage error, or in other words the following quantity:

$$\frac{100}{n \cdot k} \sum_{j=1}^n \sum_{i=1}^k \left| \frac{\sigma_{mkt}(K_i, T_j) - \sigma_{model}(K_i, T_j)}{\sigma_{mkt}(K_i, T_j)} \right|$$

Here we report the implied volatility surface calibrated to all the strikes and tenors together:



where the parameters are:

$$\eta = 0.0568 \quad \kappa = 2.6523 \quad \theta = 1.3231 \quad \rho = -0.6766 \quad \sigma_0 = 0.0442$$

the mean relative percentage error obtained is 4.5817% and it took 4.2 s to calibrate on a normal laptop. The calibration procedure is quite efficient, however we want to highlight two facts:

1. the parameters obtained do not satisfy the Feller condition, which we remind that it is a sufficient condition to have the variance process always positive, but not necessary;
2. we are underestimating a bit the implied volatility for the shorter tenors.

## 6 Simulation

Suppose now that we have calibrated the model and we wanted to sample paths from it, then we need an integration scheme. Broadie and Kaya developed an exact and bias-free scheme to sample Heston's paths, however it has a major drawback: it is incredibly slow for practical applications. To circumvent this problem many practitioners use variation of the Euler scheme or the Milstein scheme, which are classical scheme for numerically integrating SDEs. These schemes need a correction in order to keep the variance process positive, the correction can be that if the variance at a certain time step becomes negative then it is put equal to zero. They require small timesteps and have worse convergence than the Broadie-Kaya's one, moreover whenever the Feller's condition is violated the results are not reliable. We chose to present the Gamma Approximation scheme proposed by Jean-François Bégin, Mylène Bédard and Patrice Gaillardetz, which has the advantages to be almost as accurate as the Broadie-Kaya's one, to be computationally faster and to be low-bias. Before starting, in the following sections we will use the Heston model w.r.t. the log-prices  $X_t = \log(S_t)$ . Using Itô's formula we can rewrite, under risk-neutral measure, the model as:

$$\begin{cases} dX_t = \left( r - q - \frac{1}{2}v_t \right) dt + \sqrt{v_t} dW_t \\ dv_t = \kappa(\eta - v_t)dt + \theta\sqrt{v_t} d\tilde{W}_t \\ v_0 = \sigma_0^2 \end{cases}$$

### 6.1 Cumulative distribution function of $v_T|v_t$

We state here an important result (proven in Dufresne, "The integrated square-root process") that we will use later about the cumulative distribution function of  $v_T$  given  $v_t$ .



**Proposition 3.** Let  $F_{\chi^2}(z; \nu, \lambda)$  be the cdf of the non-central chi-square distribution with non centrality parameter  $\lambda$  and  $\nu$  degrees of freedom,

$$F_{\chi^2}(z; \nu, \lambda) = e^{-\lambda/2} \sum_{j=0}^{\infty} \frac{(\lambda/2)^j}{j! 2^{\nu/2+j} \Gamma(\nu/2 + j)} \int_0^z x^{\nu/2+j-1} e^{-x/2} dx$$

Let

$$\bar{\nu} = \frac{4\kappa\eta}{\theta^2}$$

and for  $t < T$ :

$$n(t, T) = \frac{4\kappa e^{-\kappa(T-t)}}{\theta^2 [1 - e^{-\kappa(T-t)}]}$$

then we have that

$$\mathbf{Pr}_{\mathbb{Q}}(v_T < x | v_t) = F_{\chi^2}\left(\frac{x \cdot n(t, T)}{e^{-\kappa(T-t)}}; \bar{\nu}, v_t \cdot n(t, T)\right)$$

## 6.2 Explicit solution for the log-asset price

Let  $\Delta t > 0$  be our timestep and discretize time as  $0, \Delta t, 2\Delta t, \dots, j\Delta t, \dots, T$ . Integrating the log-price equation in the Heston model and using the Cholesky's decomposition in the Brownian motion we obtain:

$$\begin{aligned} X_{j\Delta t} &= X_{(j-1)\Delta t} + (r - q)\Delta t - \frac{1}{2} \int_{(j-1)\Delta t}^{j\Delta t} v_t dt \\ &\quad + \rho \int_{(j-1)\Delta t}^{j\Delta t} \sqrt{v_t} d\tilde{W}_t + \sqrt{1 - \rho^2} \int_{(j-1)\Delta t}^{j\Delta t} \sqrt{v_t} d\tilde{W}_t^\perp \end{aligned}$$

and in the same way integrating the variance process yields:

$$V_{j\Delta t} = V_{(j-1)\Delta t} + \int_{(j-1)\Delta t}^{j\Delta t} \kappa[\eta - v_t] dt + \theta \int_{(j-1)\Delta t}^{j\Delta t} \sqrt{v_t} d\tilde{W}_t$$

Since  $\theta > 0$  we can isolate the integral in the last term as:

$$\int_{(j-1)\Delta t}^{j\Delta t} \sqrt{v_t} d\tilde{W}_t = \theta^{-1} \left[ v_{j\Delta t} - v_{(j-1)\Delta t} - \kappa\eta\Delta t + \kappa \int_{(j-1)\Delta t}^{j\Delta t} v_t dt \right]$$

and substituting into the first equation of this section we obtain:

$$\begin{aligned} X_{j\Delta t} &= X_{(j-1)\Delta t} + (r - q)\Delta t - \frac{1}{2} \int_{(j-1)\Delta t}^{j\Delta t} v_t dt + \frac{\kappa\rho}{\theta} \int_{(j-1)\Delta t}^{j\Delta t} v_t dt \\ &\quad + \frac{\rho}{\theta} [v_{j\Delta t} - v_{(j-1)\Delta t} - \kappa\eta\Delta t] + \sqrt{1 - \rho^2} \int_{(j-1)\Delta t}^{j\Delta t} \sqrt{v_t} d\tilde{W}_t^\perp \end{aligned}$$

we define the integrated variance as:

$$IV_{(j-1)\Delta t}^{j\Delta t} := \int_{(j-1)\Delta t}^{j\Delta t} v_t dt$$

using this notation and the equation above and approximating the stochastic integral, Brodie and Kaya obtained the following equation:

$$\begin{aligned} \hat{X}_{j\Delta t} = \hat{X}_{(j-1)\Delta t} + (r - q)\Delta t - \frac{1}{2}I\hat{V}_{(j-1)\Delta t}^{j\Delta t} + \frac{\kappa\rho}{\theta}I\hat{V}_{(j-1)\Delta t}^{j\Delta t} + \\ \frac{\rho}{\theta}[\hat{v}_{j\Delta t} - \hat{v}_{(j-1)\Delta t} - \kappa\eta\Delta t] + Z\sqrt{1 - \rho^2}\sqrt{I\hat{V}_{(j-1)\Delta t}^{j\Delta t}} \end{aligned} \quad (8)$$

where  $Z \sim \mathcal{N}(0, 1)$ .

### 6.3 Gamma approximation scheme

The algorithm works as follow:

---

**Algorithm 2** GA Scheme

---

- 1: Create caches for the moments of  $IV_{(j-1)\Delta t}^{j\Delta t}$  as explained in section 6.4.
  - 2: Sample  $\hat{v}_{j\Delta t}$  given  $\hat{v}_{(j-1)\Delta t}$  from the non-central chi-square distribution described in section 6.1.
  - 3: Given  $\hat{v}_{j\Delta t}$  and  $\hat{v}_{(j-1)\Delta t}$ , calculate the integrated variance over time,  $I\hat{V}_{(j-1)\Delta t}^{j\Delta t}$  from a moment-matched gamma distribution using the moments available in the caches.
  - 4: Sample  $Z \sim \mathcal{N}(0, 1)$  and use equation (8) to obtain  $\hat{X}_{j\Delta t}$ .
- 

the fact that the caches can be precomputed is what make this algorithm so computationally efficient.

### 6.4 Implementation of caches

Before discussing how the caches are implemented we need two technical propositions.

**Proposition 4.** *The integrated variance admits the representation*

$$IV_t^T \stackrel{d}{=} X_1 + X_2 + \sum_{j=1}^{\xi} Z_j$$

where  $X_1, X_2, Z_j \forall j$  and  $\xi$  are mutually independent. The random variables

$X_1, X_2$  and  $Z_j$  have the following representations:

$$\begin{aligned} X_1 &\stackrel{d}{=} \sum_{n=1}^{\infty} \frac{1}{\gamma_n} \sum_{j=1}^{N_n} A_j \\ X_1 &\stackrel{d}{=} \sum_{n=1}^{\infty} \frac{1}{\gamma_n} B_n \\ Z_j &\stackrel{d}{=} \sum_{n=1}^{\infty} \frac{1}{\gamma_n} C_{n,j} \end{aligned}$$

where

$$\gamma_n = \frac{\kappa(T-t)^2 + 4\pi^2 n^2}{2\theta^2(T-t)^2}$$

Here, the  $A_j$  are independent exponential random variables with mean 1,  $N_n$  are independent Poisson random variables with respective means  $(v_t + v_T)\lambda_n$  and

$$\lambda_n = \frac{16\pi^2 n^2}{\theta^2(T-t)[\kappa^2(T-t)^2 + 4\pi^2 n^2]}$$

The  $B_n$  are independent gamma random variables with a shape parameter of  $\bar{\nu}/2$  and a scale parameter of 1. Finally,  $\xi$  is a Bessel random variable with parameter

$$z = \frac{2\kappa}{\theta^2 \sinh[\kappa(T-t)/2]} \sqrt{v_t v_T}$$

and degrees of freedom equal to  $\xi/2 - 1$ .

**Proposition 5.** Let  $C_1 = \coth[\kappa(T-t)/2]$  and  $C_2 = \operatorname{csch}^2[\kappa(T-t)/2]$ . Given  $v_t$  and  $v_T$  the mean and variance of  $IV_t^T$  are expressed as

$$\mathbb{E}[IV_t^T | v_t, v_T] = \mathbb{E}[X_1] + \mathbb{E}[X_2] + \mathbb{E}[\xi] \mathbb{E}[Z]$$

and

$$\operatorname{Var}[IV_t^T | v_t, v_T] = \operatorname{Var}[X_1] + \operatorname{Var}[X_2] + \mathbb{E}[\xi] \operatorname{Var}[Z] + (\mathbb{E}[\xi^2] - \mathbb{E}[\xi]^2) \mathbb{E}[Z]^2$$

the mean and variance of  $X_1$  respectively satisfy

$$\begin{aligned} \mathbb{E}[X_1] &= (v_t + v_T) \left[ \frac{C_1}{\kappa} - (T-t) \frac{C_2}{2} \right] \\ \operatorname{Var}[X_1] &= (v_t + v_T) \theta^2 \left[ \frac{C_1}{\kappa^3} + (T-t) \frac{C_2}{2\kappa^2} - (T-t)^2 \frac{C_1 C_2}{2\kappa} \right] \end{aligned}$$

the mean and variance of  $X_2$  respectively satisfy

$$\begin{aligned} \mathbb{E}[X_2] &= \bar{\nu} \theta^2 \left[ \frac{-2 + \kappa(T-t)C_1}{4\kappa^2} \right] \\ \operatorname{Var}[X_2] &= \bar{\nu} \theta^4 \left[ \frac{-8 + 2\kappa(T-t)C_1 + \kappa^2(T-t)^2 C_2}{8\kappa^4} \right] \end{aligned}$$

the mean and the variance of  $Z$  respectively satisfy

$$\begin{aligned}\mathbb{E}[Z] &= 4\mathbb{E}[X_2]/\bar{\nu} \\ \text{Var}[Z] &= 4\text{Var}[X_2]/\bar{\nu}\end{aligned}$$

Finally,

$$\begin{aligned}\mathbb{E}[\xi] &= \frac{z\mathcal{I}_{\bar{\nu}/2}(z)}{2\mathcal{I}_{\bar{\nu}/2-1}(z)} \\ \mathbb{E}[\xi^2] &= \mathbb{E}[\xi] + \frac{z^2\mathcal{I}_{\bar{\nu}/2+1}(z)}{4\mathcal{I}_{\bar{\nu}/2-1}(z)}\end{aligned}$$

where  $z$  is as in the previous proposition and  $\mathcal{I}_\nu(\cdot)$  is a Bessel function of the first kind with  $\nu$  degrees of freedom.

With these two proposition we can now proceed and define  $IV_{(j-1)\Delta t}^{*j\Delta t} := IV_{(j-1)\Delta t}^{j\Delta t} - X_1$  using the same notation as in Proposition 4. Now we obtain:

$$\begin{aligned}\mathbb{E}[IV_{(j-1)\Delta t}^{j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] &= \mathbb{E}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] + \mathbb{E}[X_1] \\ \text{Var}[IV_{(j-1)\Delta t}^{j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] &= \text{Var}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] + \text{Var}[X_1]\end{aligned}$$

And since the first two moments of  $X_1$  depend only on  $v_{(j-1)\Delta t} + v_{j\Delta t}$  and do not require the evaluation of any Bessel function, can be computed later in an inexpensive fashion. The moments computation will be performed in the following way:

---

**Algorithm 3** IV Moment Computation

---

- 1: Precompute in some predefined points (called *totems*)  $\mathbb{E}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$  and  $\text{Var}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$ . These values are called *caches*.
  - 2: Compute  $\mathbb{E}[X_1]$  and  $\text{Var}[X_1]$ .
  - 3: Use linear interpolation of caches to approximate  $\mathbb{E}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$  and  $\text{Var}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$ .
  - 4: Add  $\mathbb{E}[X_1]$  and  $\text{Var}[X_1]$  to the previous moments respectively to obtain  $\mathbb{E}[IV_{(j-1)\Delta t}^{j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$  and  $\text{Var}[IV_{(j-1)\Delta t}^{j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$ .
- 

Notice that  $\mathbb{E}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$  and  $\text{Var}[IV_{(j-1)\Delta t}^{*j\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$  are functions of  $v_{(j-1)\Delta t} \times v_{j\Delta t}$  and with respect to this quantity they display an exponential behaviour. So the natural grid (way to choose the totems) is to choose an exponentially spaced grid.

## 7 At-the-money forward skew

We can define the at-the-money forward skew for options with tenor  $T$  as

$$\psi(T) = S_0 e^{(r-q)T} \cdot \left| \frac{\partial \sigma_{mkt}(K, T)}{\partial K} \right|_{K=S_0 e^{(r-q)T}}$$

where  $\sigma_{mkt}(K, T)$  is the implied volatility observed in the market for options with strike  $K$  and tenor  $T$ . In order to compute this quantity we need to fit a sufficiently precise model which will almost interpolate the implied vols observed in the market. To achieve that we will fit a stochastic volatility inspired model presented in *Arbitrage-free SVI volatility surfaces* by Gatheral and Jacquier. Without entering too much into the details the model is the following, where  $k$  is the forward log-strike,

$$\sigma_{SVI}^2(k) = a_T + b_T \left[ \rho_T(k - m_T) + \sqrt{(k - m_T)^2 + \sigma_T^2} \right]$$

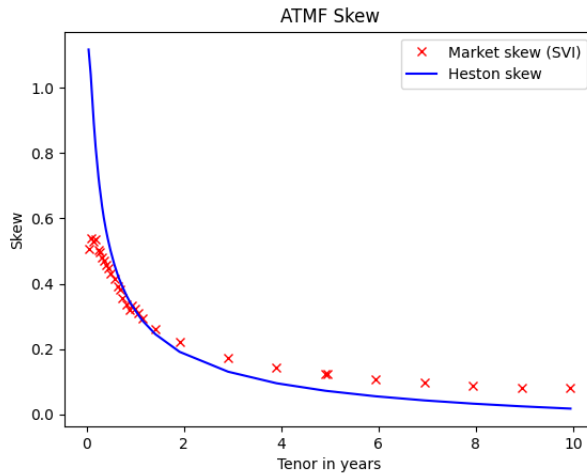
where  $a_T \in \mathbb{R}$ ,  $b_T \geq 0$ ,  $|\rho_T| < 1$ ,  $m_T \in \mathbb{R}$  and  $\sigma_T > 0$  are parameters different for each tenor. The constraint  $a_T + b_T \sigma_T \sqrt{1 - \rho_T^2} \geq 0$  guarantees that the implied volatility surface obtained is always positive. Each one of these parameters has a distinct effect on the implied volatility surface, in order to discover more please refer to the paper. Under this model we have that

$$\psi(T) = \frac{b_T}{2\sigma_{SVI}(0)} \left[ -\frac{m_T}{\sqrt{m_T^2 + \sigma_T^2}} + \rho_T \right]$$

meanwhile we can estimate the at-the-money forward skew of Heston using the central finite difference approximation. Letting  $h > 0$  be the difference step and  $T$  be one of the tenors, the approximation of  $\psi$  is given by:

$$\psi(T) \approx \left| \frac{\hat{\sigma}_H(h, T) - \hat{\sigma}_H(-h, T)}{2h} \right|$$

where  $\hat{\sigma}_H(h, T)$  is the implied volatility given by the calibrated Heston model for a vanilla option of log-forward strike  $h$ . We obtained the following result:



as we can see we have not obtained a good fit. We can notice that as  $T \rightarrow \infty$  the skew of the Heston model seems to decay faster than the market skew. This is not a perception, indeed it is observed empirically that market ATM skew decays like  $1/\sqrt{T}$  and the Heston skew decays at  $1/T$ . This is actually a big deal for practitioners, since when you are trading exotic derivatives the trader will charge either on skew or on vol spread, depending on the structure.

## References

1. Gatheral. The volatility surface. A practitioner's guide. Wiley, 2006.
2. Fang and Oosterlee. Cosine expansions for option pricing. AMS, 2008.
3. Albrecher, Hansjoerg and Mayer, Philipp and Schoutens, Wim and Tistaert, Jurgen. The Little Heston Trap. Wilmott, 2007.
4. Burzoni. Lecture notes for the course: Mathematical Finance 2, 2022.
5. Le Floc'h. More Robust Pricing of European Options Based on Fourier Cosine Series Expansions. ArXiv, 2020.
6. Cui, del Bano Rollin, Germano. Full and fast calibration of the Heston stochastic volatility model. ArXiv, 2016.
7. Bégin, Bédard and Gaillardetz. Simulating from the Heston Model: A Gamma Approximation Scheme. Monte Carlo Methods and Applications, 2014.
8. Broadie, Kaya. Exact Simulation of Stochastic Volatility and Other Affine Jump Diffusion Processes. Operations Research, 2006.
9. Gatheral, Jacquier. Arbitrage-free SVI volatility surfaces. ArXiv, 2013.

## Part III

# Rough Heston

As we have seen the skew of the at the money options in the Heston model does not represent well the skew of real-world at the money options. This, among other reasons, made the researcher and practitioner to think about more sophisticated models. As observed by Gatheral, Jaisson and Rosenbaum the log-volatility in the market behaves like a fractional Brownian motion with small Hurst parameter, so the volatility is rough.

## 1 Stylized empirical facts

Heston model reproduces several important features of low frequency price data, provides quite reasonable dynamics for the volatility surface and it can be calibrated efficiently. If we want to surpass that we have to build a model which can reproduce the stylized facts of modern electronic markets in the context of high frequency trading. In practice each market behaves tick-by-tick, indeed we receive an update in price by market-makers whenever there is a trade and the movement is discrete and at least of one tick (usually 1 cent). There are 4 main stylized facts that we can observe in market data:

1. Markets are highly endogenous, as showed by Bouchad. This means that most of the orders have no real economic motivation, but are simply the reaction of algorithms to other orders.
2. Markets at high frequency are much more efficient than at lower frequencies, this means that it is much more difficult to find profitable statistical arbitrage strategies.
3. There is some asymmetry in the liquidity on the bid and the ask side of the order book. Indeed, a market-maker is likely to raise the price by less following a buy order than to lower the price following the same size sell order, as seen by Brunnermeier and Pedersen. This is mostly due to the fact that hedging the first position is easier than the second and that market-makers have usually some inventory.
4. A large proportion of transaction is due to big orders, called metaorders, which are not executed at once, but split in time. Indeed, one of the most challenging part of every trading strategies is to execute it in large volumes without moving changing too much the state of the market.

## 2 Building the model

As in El Euch, Fukasawa & Rosenbaum we will start building a model from Hawkes processes, then slowly including the stylized fact mentioned in the last



paragraph and showing that the long-term dynamic of this model will lead to a rough Heston model at the macroscopic scale, in which the leverage effect is still represented.

## 2.1 Hawkes Processes

Hawkes processes are point processes which are said to be self-exciting, in the sense that the instantaneous jump-probability depends on the location of the past events. In particular we will focus on a bivariate Hawkes process,  $(N_t^+, N_t^-)_{t \geq 0}$ , where  $N_t^+$  is the number of upward jumps of one tick and  $N_t^-$  is the number of downward jumps of one tick, both in the interval  $[0, t]$ . The probability to get one-tick upward jump in a time  $dt$  is given by  $\lambda_t^+ dt$ , viceversa by  $\lambda_t^-$ . The array  $(\lambda_t^+, \lambda_t^-)$  is called intensity of the process and it is of the form:

$$\begin{pmatrix} \lambda_t^+ \\ \lambda_t^- \end{pmatrix} = \begin{pmatrix} \mu^+ \\ \mu^- \end{pmatrix} + \int_0^t \begin{pmatrix} \phi_1(t-s) & \phi_3(t-s) \\ \phi_2(t-s) & \phi_4(t-s) \end{pmatrix} \cdot \begin{pmatrix} dN_s^+ \\ dN_s^- \end{pmatrix}$$

where  $\mu^+$  and  $\mu^-$  are positive constants and the components of the matrix are positive and locally integrable functions. The process of the prices  $P_t$  is given by the difference between the number of upward jumps from time 0 and the number of downward jumps, so:

$$P_t = N_t^+ - N_t^-$$

Each component of the intensity can be decomposed into three parts, for example  $\lambda_t^-$  can be decomposed in:

- $\mu_t^-$  which corresponds to the probability that the price will go down because of exogenous reasons;
- $\int_0^t \phi_2(t-s) dN_s^+$  which corresponds to the probability of a downward jump induced by past upward jumps;
- $\int_0^t \phi_4(t-s) dN_s^-$  which corresponds to the probability of a downward jump induced by past downward jumps.

Now we will see that, when the  $\phi_j$  have suitable forms the model can reproduce the stylized effects described in the previous section. Moreover, we want to underlying that, due to how the model is built, the price process assumes discrete values, as in the real world.

## 2.2 Encoding the 2<sup>nd</sup> property

Since the markets at high frequency are expected to be more efficient then this translate in that, over any period of time, we should have on average the same number of upwards jumps than downwards jumps. This can be translated in:

$$\int_0^t \mathbb{E}[\lambda_s^+] ds = \mathbb{E}[N_t^+] = \mathbb{E}[N_t^-] = \int_0^t \mathbb{E}[\lambda_s^-] ds \quad (9)$$

remembering how we have defined  $\lambda_t^+$  and  $\lambda_t^-$ :

$$\begin{aligned}\mathbb{E}[\lambda_t^+] &= \mu^+ + \int_0^t \phi_1(t-s) \mathbb{E}[\lambda_s^+] ds + \int_0^t \phi_3(t-s) \mathbb{E}[\lambda_s^-] ds \\ \mathbb{E}[\lambda_t^-] &= \mu^- + \int_0^t \phi_4(t-s) \mathbb{E}[\lambda_s^-] ds + \int_0^t \phi_2(t-s) \mathbb{E}[\lambda_s^+] ds\end{aligned}$$

the simplest way to satisfy the equation (1) is to put:

$$\mu^+ = \mu^- \text{ and } \phi_1 + \phi_3 = \phi_2 + \phi_4$$

### 2.3 Encoding the 3<sup>rd</sup> property

Market-makers act as liquidity providers, in practice at the beginning they are long inventory, so the ask side is more liquid than the bid side. This translate into the fact that the conditional probability of an upward jump right after an upward jump is smaller than the conditional probability to observe a downward jump after a downward jump. This means that for  $t \rightarrow 0$  we have:

$$\int_0^t \phi_4(t-s) dN_s^- > \int_0^t \phi_1(t-s) dN_s^+$$

or equivalently

$$\int_0^t \phi_2(t-s) dN_s^+ < \int_0^t \phi_3(t-s) dN_s^-$$

this can be satisfied in several ways, but we make the strong assumption that exists a constant  $\beta > 0$  such that  $\phi_3 = \beta\phi_2$ . Putting all together we have that the structure of our intensity process is:

$$\begin{pmatrix} \lambda_t^+ \\ \lambda_t^- \end{pmatrix} = \mu \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \int_0^t \begin{pmatrix} \phi_1(t-s) & \beta\phi_2(t-s) \\ \phi_2(t-s) & [\phi_1 + (\beta-1)\phi_2](t-s) \end{pmatrix} \cdot \begin{pmatrix} dN_s^+ \\ dN_s^- \end{pmatrix}$$

### 2.4 Encoding the 1<sup>st</sup> property

Markets have an high degree of endogeneity, which means that the proportion of "non-meaningful" orders with respect to the totality of the orders is close to 1. In order to have an intuition on how to include this effect in our model we need to recall dynamical systems. A dynamical system has a stationary point (or equilibrium) if its spectral radius is less than 1, in the same way we have a kernel transition matrix:

$$\int_0^T \begin{pmatrix} \phi_{1,T}(s) & \beta\phi_{2,T}(s) \\ \phi_{2,T}(s) & [\phi_{1,T} + (\beta-1)\phi_{2,T}](s) \end{pmatrix} ds = \int_0^T \Phi_T(s) ds$$

we can extend all the functions on  $[0, \infty)$  with the constant zero. We refer to them with a tilde. The spectral radius in our case is equal to:

$$\sigma\left(\int_0^\infty \tilde{\Phi}_T(s)ds\right) = \|\tilde{\phi}_{1,T}\|_1 + \beta\|\tilde{\phi}_{2,T}\|_1$$

Let  $(\Omega, \mathbb{F} = \{(F_T)_{T \geq 0}\}, \mathbb{P})$  a complete filtered probability space. We can find a sequence  $\{(\tilde{\phi}_{1,T}; \tilde{\phi}_{2,T})\}_T$  of couple of positive functions each one in the respective  $\mathcal{L}^1(F_T)$  such that:

- $\forall T > 0$  we have  $\|\tilde{\phi}_{1,T}\|_1 + \beta\|\tilde{\phi}_{2,T}\|_1 < 1$ ;
- if  $T_2 > T_1$  we have both  $\tilde{\phi}_{1,T_2} \geq \tilde{\phi}_{1,T_1}$  and  $\tilde{\phi}_{2,T_2} \geq \tilde{\phi}_{2,T_1}$ ;
- satisfying:

$$\lim_{T \rightarrow \infty} \left[ \|\tilde{\phi}_{1,T}\|_1 + \beta\|\tilde{\phi}_{2,T}\|_1 \right] = 1$$

then there exist a limit to this sequence and we will call that  $(\tilde{\phi}_1; \tilde{\phi}_2)$  and, due to continuity of the norm, we have that  $\|\tilde{\phi}_1\|_1 + \beta\|\tilde{\phi}_2\|_1 = 1$ . Then we have built our nearly-unstable system for each  $T > 0$  sufficiently large. Moreover notice that, due to the second property of our sequence, we have that the spectral radius, when  $T$  is increasing, is also increasing. We will refer to the matrix obtained with  $(\tilde{\phi}_1; \tilde{\phi}_2)$  as  $\Phi$ . The process of the prices up to time  $T < \infty$  is now indicated as:

$$P_t^T = N_t^{T,+} - N_t^{T,-}$$

with  $N_t^{T,+}$  and  $N_t^{T,-}$  with intensity generated by  $(\tilde{\phi}_{1,T}; \tilde{\phi}_{2,T})$ . From now on, we will denote

$$\sigma\left(\int_0^\infty \tilde{\Phi}_T(s)ds\right) = a_T = \|\tilde{\phi}_{1,T}\|_1 + \beta\|\tilde{\phi}_{2,T}\|_1$$

with  $a_T$  constants and  $a_T \uparrow 1$ . Moreover, we can also construct the  $\Phi_T$  as  $\Phi = a_T \Phi_T$ . We will refer to this as **Assumption 1**.

## 2.5 Encoding the 4<sup>th</sup> property

As showed by Jaisson and Rosenbaum, the effect of metaorders are reflected in the Hawkes framework by considering the condition that the kernel matrix exhibits heavy-tails. In order to encode the metaorders in the framework we need to put some additional assumptions. Let

$$\Psi_T = \sum_{k \geq 1} (\Phi_T)^{*k}$$

where  $(\Phi_T)^{*1} = \Phi_T$  and, for  $k > 1$ ,  $(\Phi_T)^{*k}(t) = \int_0^t \Phi_T(s)(\Phi_T)^{*(k-1)}(t-s)ds$ . The **Assumption 2** is that  $\Psi_T$  is uniformly bounded,  $\Phi$  is differentiable and the derivative of each component of  $\Phi$  is bounded and with finite norm 1. In

order to satisfy this assumption is sufficient, but not necessary, that  $\phi_1$  and  $\phi_2$  are both non-increasing functions in  $\mathcal{L}^1 \cap \mathcal{L}^\infty$  and differentiable. The last assumption, **Assumption 3**, is that there exist  $\alpha \in (1/2, 1)$  and  $C > 0$  such that

$$\alpha t^\alpha \int_t^\infty [\phi_1 + \beta \phi_2](s) ds \xrightarrow[t \rightarrow \infty]{} C$$

and moreover, for some  $\mu > 0$  and  $\lambda^* > 0$ ,

$$T^\alpha(1 - a_T) \xrightarrow[t \rightarrow \infty]{} \lambda^* \text{ and } T^{1-\alpha} \mu_T \xrightarrow[t \rightarrow \infty]{} \mu$$

under this three assumptions also the last stylized effect has been encoded in our model.

### 2.5.1 Wiener-Hopf equations

**Assumption 2** and the definition of the  $\Psi_T$  seems a little obscure. This assumption is there only because it allows us to use the following result on integral equations:

**Lemma.** *Let  $g$  be a measurable locally bounded function from  $\mathbb{R}$  to  $\mathbb{R}^2$  and  $\Phi : \mathbb{R}^+ \rightarrow \mathcal{M}^2(\mathbb{R})$  be a matrix-value function with integrable components such that the spectral radius of  $\int_0^\infty \phi(s) ds < 1$ . Then there exists a unique locally bounded function  $f$  from  $\mathbb{R}$  to  $\mathbb{R}^2$  solution of*

$$f(t) = g(t) + \int_0^t \langle \Phi(t-s), f(s) \rangle ds, \quad t \geq 0$$

given by

$$f(t) = g(t) + \int_0^t \langle \Psi(t-s), g(s) \rangle ds, \quad t \geq 0$$

where  $\Psi = \sum_{k \geq 1} \Phi^{*k}$ .

## 2.6 From microstructure to macrostructure

If **Assumptions 1,2 and 3** hold then it happens that the asymptotic behaviour of the microstructural model that we have built behaves like an Heston model, more precisely a rough version of it. Indeed, using the same notation as in **Assumption 3**, let:

$$\lambda = \frac{\alpha \lambda^*}{CT(1 - \alpha)}$$

then it holds true the following theorem:

**Teorema.** *As  $T \rightarrow \infty$  then the rescaled microscopic price*

$$\sqrt{\frac{1 - a_T}{\mu T^\alpha}} P_t^T$$

converges in the sense of finite dimensional laws to the following rough Heston model:

$$P_t = \frac{1}{1 - (\|\phi_1\|_1 - \|\phi_2\|_1)} \sqrt{\frac{2}{\beta + 1}} \int_0^t \sqrt{v_s} dW_s$$

where  $v_t$  is the solution to the following rough SDE:

$$v_t = \frac{1}{\Gamma(\alpha)} \left[ \int_0^t (t-s)^{\alpha-1} \lambda (1+\beta-v_s) ds + \int_0^t (t-s)^{\alpha-1} \lambda \sqrt{\frac{1+\beta^2}{\lambda^* \mu (1+\beta)^2}} \sqrt{v_s} d\tilde{W}_s \right]$$

where  $(W, \tilde{W})$  are two correlated Brownian motions with:

$$d\langle W, \tilde{W} \rangle_t = \frac{1-\beta}{\sqrt{2(1+\beta^2)}} dt$$

furthermore, the process  $v_t$  has Hölder regularity  $\alpha - 1/2 - \varepsilon$  for each  $\varepsilon > 0$ .

The proof is really technical and out of our scope. It can be found in El Euch, Fukasawa and Rosenbaum.

## 2.7 Mittag-Leffler functions

As we can see in the previous theorem we are implicitly assuming that  $v_0 = 0$ . Well, for a practitioner this is a severe limitation. However, it is not immediate to obtain the same result if  $v_0$  is not zero. In this subsection we will give some definitions which we will use to obtain a more general result. First of all, we define the Mittag-Leffler functions. Let  $\alpha, \beta \in \mathbb{R}^+$ , then the Mittag-Leffler function  $E_{\alpha, \beta}$  is defined for  $t \in \mathbb{C}$  and  $C \in \mathbb{C}$  as

$$E_{\alpha, \beta}(Ct^\alpha) = \sum_{j=0}^{\infty} \frac{C^j t^{j\alpha}}{\Gamma(\alpha j + \beta)}$$

if  $\alpha \in (0, 1)$  and  $\lambda \in \mathbb{R}^+$ , we can also define

$$\begin{aligned} f^{\alpha, \lambda}(t) &= \lambda t^{\alpha-1} E_{\alpha, \alpha}(-\lambda t^\alpha) \mathbf{1}_{t \geq 0} \\ F^{\alpha, \lambda}(t) &= \int_0^t f^{\alpha, \lambda}(s) ds \end{aligned}$$

the first one can be proven to be a density function on  $\mathbb{R}^+$  and it is called Mittag-Leffler density function. The Mittag-Leffler density function has many nice properties, among the others, we are interested in the followings:

$$\begin{aligned} f^{\alpha, \lambda}(t) &\sim \frac{\alpha}{\lambda \Gamma(1-\alpha)} t^{-\alpha-1} \quad \text{if } t \rightarrow \infty \\ F^{\alpha, \lambda}(t) &\sim \frac{\lambda}{\Gamma(1+\alpha)} t^\alpha \quad \text{if } t \rightarrow 0^+ \\ F^{\alpha, \lambda}(t) &\sim 1 + \frac{1}{\lambda \Gamma(1-\alpha)} t^{-\alpha} \quad \text{if } t \rightarrow \infty \\ D^\alpha[E_\alpha(Ct^\alpha) - 1] &= CE_\alpha(Ct^\alpha) \end{aligned}$$

when not specified  $\beta$  is assumed to be 1. Moreover we will use this lemma:

**Lemma.** *Let  $\alpha \in (0, 1]$ ,  $u \in \mathbb{C}$  with  $u = a + ib$  with  $a \in \mathbb{R}^+$  and  $b \in [-1/(1 - \rho^2), 0]$ . Define as  $C = \sqrt{u(u + i) - \rho^2 u^2}$ . Then for any positive integer  $p$  and  $t \in \mathbb{R}^+$  this expansion holds*

$$E_\alpha(-Ct^\alpha) = \sum_{j=1}^p \frac{(-1)^{j-1} t^{-j\alpha}}{C^j \Gamma(1 - j\alpha)} + \mathcal{O}(|Ct^\alpha|^{-1-p}) \quad \text{if } t \rightarrow \infty$$

## 2.8 Adjusting the initial volatility

Luckily, we will see that in order to adjust the initial volatility is sufficient to consider an appropriate inhomogeneous intensity for our bi-dimensional Hawkes process and an appropriate kernel matrix. Indeed, using the same notation as in the previous session, we can make a very particular choice for the  $\Phi_T$ . We will choose the following: suppose that exist  $\beta \geq 0$ ,  $\alpha \in (1/2, 1)$  and  $\lambda > 0$  such that

$$a_T = 1 - \lambda T^{-\alpha}, \quad \Phi_T(s) = a_T f^{\alpha,1}(s) \chi$$

where

$$\chi = \frac{1}{\beta + 1} \begin{pmatrix} 1 & \beta \\ 1 & \beta \end{pmatrix}$$

with this particular choice we see that **Assumption 1, 2** and **3** are satisfied. Moreover, all the first three properties are still well encoded into the model. We need also to notice that  $\chi$  is an idempotent matrix, this will be useful later when deriving the characteristic function. Now to identify the appropriate inhomogeneous intensity  $\hat{\mu}_T(\cdot)$  is far more complicated and not so useful, so we will give only the final candidate:

$$\hat{\mu}_T(t) = \mu T^{\alpha-1} + \varepsilon \mu T^{\alpha-1} \left[ \frac{1 - \int_0^t a_T f^{\alpha,1}(s) ds}{1 - a_T} - \int_0^t a_T f^{\alpha,1}(s) ds \right]$$

with  $\varepsilon, \mu$  positive constants. In the process to obtain this appropriate candidate, written in *The characteristic function of rough Heston models* of Euch and Rosenbaum, they obtained also an explicit form for  $\Psi_T$  as

$$\Psi_T(Tt) = \frac{a_T f^{\alpha,\lambda}(t)}{T(1 - a_T)}$$

which will be useful in the derivation of the characteristic function. We now need to define the microscopic process converging to the log-price of the rough Heston model, for a positive function  $\gamma(\cdot)$ , as

$$P_t^T = \sqrt{\frac{\gamma(t)}{2}} \sqrt{\frac{1 - a_T}{T^\alpha \mu}} (N_{tT}^{T,+} - N_{tT}^{T,-}) - \frac{\gamma(t)}{2} \frac{1 - a_T}{T^\alpha \mu} N_{tT}^{T,+}$$

and finally we have the following

**Teorema 6.** *As  $T \rightarrow \infty$ , under the assumptions made in this section, the sequence of processes  $(P_t^T)_{t \in (0,1)}$  converges in law for the Skorokhod topology to*

$$P_t = \int_0^t \sqrt{v_s} d\tilde{W}_s - \frac{1}{2} \int_0^t v_s ds$$

where  $v$  is the unique solution of the rough stochastic differential equation

$$v_t = \gamma(t)\varepsilon + \frac{1}{\Gamma(\alpha)} \left[ \int_0^t (t-s)^{\alpha-1} \lambda(\gamma(s) - v_s) ds + \lambda \sqrt{\frac{\gamma(s)(1+\beta^2)}{\lambda\mu(1+\beta)^2}} \int_0^t (t-s)^{\alpha-1} \sqrt{v_s} dW_s \right]$$

with  $(\tilde{W}, W)$  correlated Brownian motions with

$$d\langle \tilde{W}, W \rangle_t = \frac{1-\beta}{\sqrt{2(1+\beta^2)}} dt$$

### 3 Rough Heston model

We have seen that the rough Heston model is what arise from taking the limit of Hawkes processes. Here we add also the drift term to the equation and substitute  $\alpha$  with  $H + 1/2$ . Given a stock price process  $S = (S_t)_{t \geq 0}$  the rough Heston model, under risk-free probability measure  $\mathbb{Q}$ , is the following :

$$\begin{cases} dS_t = (r - q)S_t dt + S_t \sqrt{v_t} d\tilde{W}_t \\ v_t = v_0 + \frac{\lambda}{\Gamma(H + \frac{1}{2})} \int_0^t \frac{\gamma(s) - v_s}{(t-s)^{\frac{1}{2}-H}} ds + \frac{\theta}{\Gamma(H + \frac{1}{2})} \int_0^t \frac{\sqrt{v_s}}{(t-s)^{\frac{1}{2}-H}} dW_s \end{cases}$$

where:

- $r$  is the risk-free rate;
- $q$  is the yield of the underlying;
- $W$  and  $\tilde{W}$  are correlated Brownian motions with  $d\langle \tilde{W}, W \rangle_t = \rho dt$ ;
- $H \in (0, 1/2)$  is the Hurst exponent of the fractional Brownian motion;
- $\theta$  is the volatility of the volatility;
- $\lambda \geq 0$  is a constant representing the "speed" of the mean reversion;
- $\gamma(\cdot)$  is a positive  $F_0$ -measurable function representing the mean reversion level for the volatility.

In order to proceed with this chapter we need to define the fractional integral and the fractional derivative. We define the fractional integral of order  $\alpha \in (0, 1]$  of a function  $f$  as

$$I^\alpha f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} f(s) ds$$

whenever the integral exists. We define the fractional derivative of order  $\alpha \in (0, 1]$  of a function  $f$  as

$$D^\alpha f(t) = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \int_0^t (t-s)^{-\alpha} f(s) ds$$

whenever it exists.

### 3.1 Inference of $\lambda\gamma(\cdot)$ from the forward variance curve

A variance swap with maturity  $T$  is a contract which pays out the realized variance of a financial underlying, computed as the sum of the squares of daily log-returns, in exchange for a fixed strike called the variance swap variance  $V_0^T$  that is determined in such a way that the initial value of the contract is zero. In practice can be seen as a way to trade implied volatility with future realized volatility. We can define the volatility of a variance swap at time zero as:

$$\hat{\sigma}_0^T := \sqrt{\frac{V_0^T}{T}}$$

Using the definition the volatility of the swap at time  $t$  is

$$(\hat{\sigma}_t^T)^2 = \frac{1}{T-t} \mathbb{E} \left[ \int_t^T v_s ds \middle| F_t \right] = \frac{1}{T-t} \int_t^T \xi_t(s) ds$$

where  $\xi_t(s) := \mathbb{E}[v_s | F_t]$  (with  $s > t$ ) and it is called the forward variance curve. So equivalently we can derive  $\xi_0(\cdot)$  as

$$\xi_0(t) = (\hat{\sigma}_0^t)^2 + t \frac{d}{dt} [(\hat{\sigma}_0^t)^2]$$

In *Perfect hedging in rough Heston models* El Euch and Rosenbaum showed that there is a link between  $\lambda\gamma(\cdot)$  and the forward variance curve. Assume that the curve admits a fractional derivative of order  $\alpha$ , then  $\lambda\gamma(\cdot)$  can be chosen so that the model is consistent with the market observed forward variance curve by taking

$$\lambda\gamma(t) = D^\alpha [\xi_0(\cdot) - v_0](t) - \lambda\xi_0(t) \quad (10)$$

Equation (2) can be obtained by calculating the conditional expected value of  $v_t$  using the second equation of the rough Heston model, showing that the forward variance curve is locally integrable (so the expected value of the stochastic integral is zero), then fractionally differentiate the LHS and RHS and reorder.



Using this fact and assuming that  $\lambda$  is sufficiently small, we can rewrite the dynamic in a compact way as:

$$(\star) \quad \begin{cases} dS_t = (r - q)S_t dt + S_t \sqrt{v_t} \{ \rho dW_t + \sqrt{1 - \rho^2} dW_t^\perp \} \\ v_t = \xi_0(t) + \frac{\theta}{\Gamma(H + \frac{1}{2})} \int_0^t \frac{\sqrt{v_s}}{(t - s)^{\frac{1}{2} - H}} dW_s \end{cases}$$

the hypothesis that  $\lambda$  must be sufficiently small is sensible since the volatility is *slowly* mean reverting.

## 4 The Characteristic function

Let  $x_t$  be the log-spot price, we wish to obtain the characteristic function of terminal log-spot  $x_T$  conditional on the initial log-price  $x_0$  and the initial forward variance curve  $\xi_0(\cdot)$ . In mathematical terms:

$$\phi_{rH}(u, T; 0) = \mathbb{E}_{\mathbb{Q}}[e^{iu x_T} | x_0, \xi_0(\cdot)]$$

### 4.1 The Characteristic function of an Hawkes process

We need to do a step back and restart from the 2-dimensional Hawkes process presented in the second section of this chapter, called  $N$ . Using the same notation as before, let  $L(u, t)$  be the characteristic function of the 2-dimensional Hawkes process  $N$  conditional at time  $t$ :

$$L(u, t) = \mathbb{E}[e^{i\langle u, N_t \rangle}], \quad u \in \mathbb{R}^2$$

then

**Teorema 7.** *We have*

$$L(u, t) = \exp \left\{ \int_0^t \langle C(u, t - s) - (1, 1), \mu(s) \rangle ds \right\}$$

where  $C : \mathbb{R}^2 \times \mathbb{R}^+ \rightarrow \mathbb{C}^2$  is the solution of this integral equation:

$$C(u, t) = \exp \left\{ iu + \int_0^t \Phi_T^\top(s) [C(u, t - s) - (1, 1)] ds \right\}$$

In order to prove this statement we define two auxiliary independent 2 dimensional point processes  $(\tilde{N}_j)_{j=1,2}$ . We will refer to  $\tilde{N}_1^2$  for the second component of the first auxiliar process, in the same way to the others. Let  $(\tilde{N}_j^k)_{j=1,2}$  be two bivariate Hawkes processes with kernel matrix  $\Phi : \mathbb{R}^+ \rightarrow \mathcal{L}^2(\mathbb{R}^+) \cap \mathcal{L}^1(\mathbb{R}^+)$ . We will denote their characteristic function at time  $t$  as  $L_j(u, t)$ . Now, for each  $j$  and each  $t \geq 0$ , let  $N_t^{0,1}$  be the number of upwards jumps occurred up to time  $t$  and  $N_t^{0,2}$  be the number of downwards jumps occurred up to time  $t$ , each one is a Poisson process with rates, respectively,  $\mu_1(t)$  and  $\mu_2(t)$ . We also define as

$\tau_1^k < \dots < \tau_{N_t^{0,k}}^k \in [0, t]$  the arrival times of jumps of type  $k$  (where  $k = 1$  for up,  $k = 2$  for down) of  $N$  up to time  $t$ . The number of jumps of type  $k$  arrived at time  $\tau_u^k$  has the same law as  $(\tilde{N}_{j,t-\tau_u^k}^k)_{j=1,2}$  where  $\tilde{N}$  is taken independent from  $N$ . So we can write the following equality, in law:

$$N_t^k = N_t^{0,k} + \sum_{j=1}^2 \sum_{l=1}^{N_t^{0,j}} \tilde{N}_{j,t-\tau_l^j}^{k,l}$$

where  $(\tilde{N}_j^{k,l})_{j=1,2}$  are  $l$  independent copies of  $(\tilde{N}_j^k)_{j=1,2}$ , also independent of  $(N^{0,k})$ . Using these considerations we can obtain:

$$\mathbb{E}[e^{i\langle u, N_t \rangle} | N_t^0] = e^{i\langle u, N_t^0 \rangle} \prod_{j=1}^2 \prod_{l=1}^{N_t^{0,j}} L_j(u, t - \tau_l^j)$$

Now, fixing  $k$ , remember that, conditional on  $N^{0,k}, t$ , we have that the vector of the arrival times has the same law as the order statistics  $(X_{(1)}, \dots, X_{(N^{0,k}, t)})$  built from iid variables  $(X_1, \dots, X_{N_t^{0,k}})$  with density with support in  $[0, t]$ . We will refer to that density as  $\frac{\mu_k(s) \mathbf{1}_{s \leq t}}{\int_0^t \mu_k(s) ds}$ . So, if we use the fact that the  $X_i$  are iid we obtain:

$$\mathbb{E}[e^{i\langle u, N_t \rangle} | N_t^0] = e^{i\langle u, N_t^0 \rangle} \prod_{j=1}^2 \left[ \left( \int_0^t L_j(u, t-s) \frac{\mu_j(s)}{\int_0^t \mu_j(r) dr} ds \right)^{N_t^{0,j}} \right]$$

using again independence and rearranging we obtain:

$$L(u, t) = \exp \left\{ \sum_{j=1}^2 \int_0^t (e^{iu_j} L_j(u, t-s) - 1) \mu_j(s) ds \right\}$$

using the same trick and remembering that also  $(\tilde{N}_j^k)_{j=1,2}$  are bivariate Hawkes processes with kernel matrix  $\Phi$  we can write:

$$L_k(u, t) = \exp \left\{ \sum_{j=1}^2 \int_0^t (e^{iu_j} L_j(u, t-s) - 1) \Phi_{j,k}(s) ds \right\}$$

now it is enough to define

$$C(u, t) = \begin{pmatrix} e^{iu_1} L_1(u, t) \\ e^{iu_2} L_2(u, t) \end{pmatrix}$$

and do the substitution in the previous equations to obtain the conclusion of the proof.

## 4.2 Intuition about the result

Using what we obtained in **section 2.8**, **3.1** and **4.1** we wish to give an intuition for the following result:

**Teorema 8.** *Consider the rough Heston model as  $(\star)$  with  $\rho \in (-1/\sqrt{2}, 1/\sqrt{2}]$ . Then the characteristic function of the terminal log-spot  $x_T$  conditional on the initial state  $(x_0, \xi_0)$  is*

$$\phi_{rH}(u, T; 0) = \exp \left\{ iux_0 + iu(r - q)T + \int_0^T D^{H+1/2} h(u, T-s) \xi_0(s) ds \right\}$$

where  $h(u, \cdot)$  is the unique continuous solution of the fractional Riccati Cauchy problem

$$\begin{cases} D^{H+1/2} h(u, \cdot) = -\frac{u^2 + iu}{2} + iu\theta\rho h(u, \cdot) + \frac{\theta^2}{2} h^2(u, \cdot) \\ I^{1/2-H} h(u, 0) = 0 \end{cases}$$

**NOTE:** the assumption that  $\rho \in (-1/\sqrt{2}, 1/\sqrt{2}]$  is simply because we have changed parameters

$$\frac{1 - \beta}{\sqrt{2(1 + \beta^2)}} := \rho$$

with  $\beta > 0$  and we need to use **Theorem 1**.

As written in **section 2.8** and remembering the definition of  $a_T$  we have that

$$P_t^T = \sqrt{\frac{\lambda\gamma(t)}{2\mu}} T^{-\alpha} (N_{tT}^{T,+} - N_{tT}^{T,-}) - \frac{\lambda\gamma(t)}{2\mu} T^{-2\alpha} N_{tT}^{T,+}$$

and we know that, if  $T \rightarrow \infty$ , this sequence of processes converges in law to  $P$  where  $P_t = \log(S_t/S_0)$ . Now let  $N^T = (N^{T,+}, N^{T,-})$  be a sequence of two dimensional Hawkes processes (varying  $T$ ) satisfying all the assumptions in **section 2.8** and denote with  $L^T(u, t)$  the characteristic function of the process  $N^T$  at point  $u = (u^+, u^-) \in \mathbb{C}^2$  and time  $t$ . If we fix a scalar  $\bar{u} \in \mathbb{C}$  and let it be

$$\bar{u}_+^T = \bar{u} \sqrt{\frac{\lambda\gamma(tT)}{2\mu}} T^{-\alpha} - \bar{u} \frac{\lambda\gamma(tT)}{2\mu} T^{-2\alpha}, \quad \bar{u}_-^T = -\bar{u} \sqrt{\frac{\lambda\gamma(tT)}{2\mu}} T^{-\alpha}$$

then we have that, since convergence in law implies pointwise convergence of the characteristics we have that

$$L^T((\bar{u}_+^T, \bar{u}_-^T), tT) \rightarrow L(\bar{u}, t) \quad \text{if } T \rightarrow \infty \quad (11)$$

where  $L(\cdot, t)$  is the characteristic function of  $P$  at time  $t$ . Now, we have to notice the following fact, which will be useful later: taking the definition of  $\hat{\mu}(\cdot)$

given in **section 2.8** and the asymptotic properties of the Mittag-Leffler density function given in **section 2.7** we may write, for each  $t \in (0, 1]$ , that

$$\begin{aligned} T^{1-\alpha} \hat{\mu}(tT) &= T^{1-\alpha} \mu_T + \varepsilon T^{1-\alpha} \mu_T \left[ \frac{T^\alpha}{\lambda} \int_{tT}^{\infty} f^{\alpha,1}(s) ds + \lambda T^{-\alpha} \int_0^{tT} f^{\alpha,1}(s) ds \right] \\ &= \mu_T \left[ 1 + \frac{\varepsilon t^{-\alpha}}{\lambda} \cdot (tT)^\alpha \int_{tT}^{\infty} f^{\alpha,1}(s) ds \right] + \mu_T \varepsilon \lambda T^{-\alpha} \int_0^{tT} f^{\alpha,1}(s) ds \\ &\xrightarrow{T \rightarrow \infty} \mu + \frac{\mu \varepsilon t^{-\alpha}}{\lambda \Gamma(1-\alpha)} \end{aligned}$$

Thanks to **Theorem 2.** we can write the characteristic function of our Hawkes process as

$$\begin{aligned} L^T((\bar{u}_+^T, \bar{u}_-^T), tT) &= \exp \left\{ \int_0^{tT} \hat{\mu}_T(s) [C^{T,+}((\bar{u}_+^T, \bar{u}_-^T), tT - s) - 1] \right. \\ &\quad \left. + C^{T,-}((\bar{u}_+^T, \bar{u}_-^T), tT - s) - 1] ds \right\} \end{aligned}$$

where  $C^T((\bar{u}_+^T, \bar{u}_-^T), t) = (C^{T,+}((\bar{u}_+^T, \bar{u}_-^T), t), C^{T,-}((\bar{u}_+^T, \bar{u}_-^T), t))$  is the solution to its respective integral equation written in **Theorem 2.** Now we define

$$Y^T(\bar{u}, t) = (Y^{T,+}(\bar{u}, t), Y^{T,-}(\bar{u}, t)) = C^T((\bar{u}_+^T, \bar{u}_-^T), tT)$$

and we can rewrite the characteristic function as

$$\begin{aligned} L^T((\bar{u}_+^T, \bar{u}_-^T), tT) &= \exp \left\{ \int_0^t T^\alpha [(Y^{T,+}(\bar{u}, t-s) - 1) + \right. \\ &\quad \left. (Y^{T,-}(\bar{u}, t-s) - 1)] \cdot [T^{1-\alpha} \hat{\mu}(sT)] ds \right\} \end{aligned}$$

Since we have (3) we can expect that as  $T \rightarrow \infty$  then  $T^\alpha(Y^T(\bar{u}, t) - (1, 1))$  converges to some functions  $(c(\bar{u}, t), d(\bar{u}, t))$ , this can be shown as in the last section of *The characteristic function of rough Heston models* of El Euch and Rosenbaum. Using the fact that  $(Y^T(\bar{u}, t) - (1, 1)) = \mathcal{O}(T^{-\alpha})$  (in the sense component-wise) we can expand  $\log(Y^T(\bar{u}, t))$  around  $(1, 1)$  (where  $\log(\cdot)$  has been applied on each component) and obtain

$$\log(Y^T(\bar{u}, t)) = Y^T(\bar{u}, t) - (1, 1) - \frac{1}{2}(Y^T(\bar{u}, t) - (1, 1))^2 + o(T^{-2\alpha})(t)$$

and using the characteristic function above and solving for  $Y^T(\bar{u}, t) - (1, 1)$  we

obtain

$$\begin{aligned}
Y^T(\bar{u}, t) - (1, 1) &= i\bar{u}\sqrt{\frac{\lambda\gamma(t)}{2\mu}}(1, -1)T^{-\alpha} - i\bar{u}\frac{\lambda\gamma(t)}{2\mu}(1, 0)T^{-2\alpha} \\
&+ T \int_0^t \phi_T^\top(Ts)(Y^T(\bar{u}, t-s) - (1, 1))ds \\
&+ \frac{1}{2}(Y^T(\bar{u}, t) - (1, 1))^2 + o(T^{-2\alpha})(t)
\end{aligned}$$

where with  $(Y^T(\bar{u}, t) - (1, 1))^2$  we intend  $\langle (Y^T(\bar{u}, t) - (1, 1)), (Y^T(\bar{u}, t) - (1, 1)) \rangle$ . Now, we wish to use the lemma in **section 2.5.1** to solve the integral part and then using the explicit form for  $\Psi_T(T\cdot)$  written in **section 2.8**. So we need to notice

$$\begin{aligned}
\sum_{k \geq 1} (T\Phi_T(T\cdot))^{*k} &= \sum_{k \geq 1} T\Phi_T^{*k}(T\cdot) \\
&= T \sum_{k \geq 1} (a_T f^{\alpha, \lambda})^{*k}(T\cdot) \chi^k \\
&= T\Psi_T(T\cdot) \chi \\
&= \frac{a_T T^\alpha}{\lambda(\beta+1)} f^{\alpha, \lambda}(\cdot) \begin{pmatrix} 1 & \beta \\ 1 & \beta \end{pmatrix}
\end{aligned}$$

and then, applying the lemma, we obtain

$$\begin{aligned}
Y^T(\bar{u}, t) - (1, 1) &= i\bar{u}\sqrt{\frac{\lambda\gamma(t)}{2\mu}}(1, -1)T^{-\alpha} - i\bar{u}\frac{a_T\gamma(t)}{2\mu(1+\beta)}(1, \beta)T^{-\alpha}F^{\alpha, \lambda}(t) \\
&+ \frac{a_T T^\alpha}{2\lambda(\beta+1)} \int_0^t f^{\alpha, \lambda}(s) \begin{pmatrix} 1 & 1 \\ \beta & \beta \end{pmatrix} (Y^T(\bar{u}, t-s) - (1, 1))^2 ds \\
&+ o(T^{-\alpha})(t)
\end{aligned}$$

so we have that  $(c(\bar{u}, t), d(\bar{u}, t))$  must satisfy the integral equations

$$\begin{aligned}
c(\bar{u}, t) &= i\bar{u}\sqrt{\frac{\lambda\gamma(t)}{2\mu}} - i\bar{u}\frac{\gamma(t)}{2\mu(1+\beta)}F^{\alpha, \lambda}(t) \\
&+ \frac{1}{2\lambda(\beta+1)} \int_0^t f^{\alpha, \lambda}(s)(c^2(\bar{u}, t-s) + d^2(\bar{u}, t-s))ds \\
d(\bar{u}, t) &= -i\bar{u}\sqrt{\frac{\lambda\gamma(t)}{2\mu}} - i\bar{u}\frac{\beta\gamma(t)}{2\mu(1+\beta)}F^{\alpha, \lambda}(t) \\
&+ \frac{\beta}{2\lambda(\beta+1)} \int_0^t f^{\alpha, \lambda}(s)(c^2(\bar{u}, t-s) + d^2(\bar{u}, t-s))ds
\end{aligned}$$

and defining  $h(\bar{u}, t) := \mu[c(\bar{u}, t) + d(\bar{u}, t)]$  then we have that

$$\begin{aligned} L(\bar{u}, t) &= \exp \left\{ \lambda \gamma(t) I^1 h(\bar{u}, t) + \varepsilon \gamma(t) I^{1-\alpha} h(\bar{u}, t) \right\} \\ &= \exp \left\{ \lambda \gamma(t) I^1 h(\bar{u}, t) + v_0 I^{1-\alpha} h(\bar{u}, t) \right\} \end{aligned}$$

where  $h$  is the solution of the fractional Riccati Cauchy problem

$$\begin{cases} D^\alpha h(u, t) = -\frac{u^2 + iu}{2} + \lambda(iu\theta\rho - 1)h(u, t) + \frac{\lambda^2\theta^2}{2}h^2(u, t) \\ I^{1-\alpha}h(u, 0) = 0 \end{cases}$$

now if we reformulate in terms of the forward variance curve, add a drift term and suppose that the starting log-spot price is not 1 we obtain the result that we wanted to prove at the beginning of this section.

### 4.3 Rational approximation of the solution

We have a quasi-closed form for the characteristic function and we wish to obtain the solution to the fractional equation. Unfortunately, this solution is not known, so we will use the Padé approximants to obtain a fast and reliable approximation. In this section we will follow the work of Gatheral and Radoicic in *Rational approximation of the rough Heston solution*. In particular we will derive an expansion for small times for the characteristic function, an expansion for long times and then we will derive a Padé rational expansion to match the two formulae.

#### 4.3.1 Small times expansion

For the small times expansion we will follow the work of Alòs *et al.*. Only for this paragraph suppose that interest rates, borrow costs and yields are zero. Let  $H(x_t, w_t(T))$  be a solution of this equation

$$-\frac{\partial H}{\partial w}(x_t, w_t(T)) + \frac{1}{2} \frac{\partial^2 H}{\partial x^2}(x_t, w_t(T)) - \frac{1}{2} \frac{\partial H}{\partial x}(x_t, w_t(T)) = 0$$

where

$$w_t(T) = \mathbb{E} \left[ \int_t^T v_s ds \middle| F_t \right] = \int_0^T \xi_0(s) ds - \int_0^t v_s ds =: M_t - \int_0^t v_s ds$$

and  $x_t$  is the log price as before. Now we need these definition

**Definition 1.** Let  $A_t$  and  $B_t$  two stochastic processes. Then

$$(A \diamond B)_t(T) := \mathbb{E} \left[ \int_t^T d\langle A_t, B_t \rangle_s \middle| F_t \right]$$

provided that the expectation is finite.

**Definition 2.** Let  $H_t := H(x_t, w_t(T))$ , defined as before, then

$$(x \diamond M)_t(T) \cdot H_t := \mathbb{E} \left[ \int_t^T d\langle x_t, M_t \rangle_s \middle| F_t \right] \frac{\partial^2 H_t}{\partial x \partial w}$$

**Definition 3.** Let  $\mathbb{F}_0 = M$ . Then the forest of order  $k \in \mathbb{N}$  is defined recursively as:

$$\mathbb{F}_k = \frac{1}{2} \sum_{l=0}^{k-2} \sum_{j=0}^{k-2} \mathbf{1}_{l+j=k-2} \mathbb{F}_l \diamond \mathbb{F}_j + x \diamond \mathbb{F}_{k-1}$$

In the last definition we dropped the subscip and the point of evaluation for  $\diamond$ , from now on, unless specified, it is always  $T$ . Using this notation Alòs, Gatheral and Radoicic obtained this powerful result

**Teorema 9.** If  $H_t$  is a solution of the differential equation presented at the beginning of this section,  $\mathbb{E}[H_T | F_t]$  is finite and for each  $j \geq 0$  the integrals in each forest  $\mathbb{F}_j$  exist. Then

$$\mathbb{E}[H_T | F_t] = e^{\sum_{j=1}^{\infty} \mathbb{F}_j} \cdot H_t$$

where the exponential is to be understood as a formal power series and  $\cdot$  is the operator in **Definition 2**.

this theorem gives us an exact representation of the conditional expectation for every model that can be written in the forward variance form without assuming Markovianity. We can now rewrite the rough Heston model in the forward variance form as:

$$\begin{cases} dS_t = S_t \sqrt{v_t} \{ \rho dW_t + \sqrt{1 - \rho^2} dW_t^\perp \} =: S_t \sqrt{v_t} d\tilde{W}_t \\ d\xi_t(u) = \frac{\theta}{\Gamma(H + \frac{1}{2})} \frac{\sqrt{v_t}}{(u - t)^{\frac{1}{2} - H}} dW_t \end{cases}$$

and, excluding the term which are  $F_t$ -measurable and do not contribute in the tree computations

$$\begin{aligned} dx_t &= \sqrt{v_t} d\tilde{W}_t + F_t\text{-measurable terms} \\ dM_t &= \frac{\theta(T - t)^\alpha}{\Gamma(\alpha + 1)} \sqrt{v_t} dW_t \end{aligned}$$

and proceeding with the computations

$$\mathbb{F}_1 = x \diamond M = \frac{\rho\theta}{\Gamma(\alpha + 1)} \int_t^T \xi_t(s) (T - s)^\alpha ds$$

if we define for  $j \in \mathbb{N}$

$$I_t^{(j)}(T) := \int_t^T \xi_t(s) (T - s)^{j\alpha} ds$$

then we have

$$\begin{aligned} dI_t^{(j)}(T) &= \int_t^T (T-s)^{j\alpha} d\xi_t(s) ds \\ &= \frac{\theta\Gamma(1+j\alpha)}{\Gamma(1+j\alpha+\alpha)} \sqrt{v_t}(T-t)^{(j+1)\alpha} dW_t + \text{drift terms} \end{aligned}$$

and in the computation of  $\diamond$  the drift terms do not contribute. With this notation we have:

$$x \diamond M = \frac{\rho\theta}{\Gamma(\alpha+1)} I_t^{(1)}(T)$$

in  $\mathbb{F}_2$  we have two trees:

$$\begin{aligned} M \diamond M &= \frac{\theta^2}{\Gamma(\alpha+1)^2} I_t^{(2)}(T) \\ x \diamond (x \diamond M) &= \frac{\rho\theta}{\Gamma(\alpha+1)} \mathbb{E} \left[ \int_t^T d\langle x, I^{(1)} \rangle_s ds \middle| F_t \right] \\ &= \frac{\rho^2\theta^2}{\Gamma(2\alpha+1)} I_t^{(2)}(T) \end{aligned}$$

it can be proven by induction that each tree in the forest  $\mathbb{F}_j$  is equal to  $\theta^j I_t^{(j)}(T)$  multiplied by a constant. Now, let's consider this characteristic function

$$H_t(u) = \phi(u, T; t) := \exp \left\{ iux_t - \frac{u^2 + ui}{2} w_t(T) \right\}$$

this clearly satisfy the PDE at the beginning of this section and, moreover, it holds true, through differentiation, that

$$e^{\sum_{j=1}^{\infty} \mathbb{F}_j} \cdot \phi(u, T; t) = e^{\sum_{j=1}^{\infty} \tilde{\mathbb{F}}_j(u)} \phi(u, T; t)$$

where  $\tilde{\mathbb{F}}_j(u)$  is defined as  $\mathbb{F}_j$  but with each occurrence of  $\partial/\partial w$  replaced with  $-(u^2 + ui)/2$  and each occurrence of  $\partial/\partial x$  replaced with  $iu$ . Using **Theorem 4.** we have that

$$\phi_{rH}(u, T; t) = \mathbb{E}_{\mathbb{Q}} [e^{iux_T} | F_t] = e^{\sum_{j=1}^{\infty} \tilde{\mathbb{F}}_j(u)} \phi(u, T; t)$$

since we have used **Theorem 4.** we have to be sure that each tree in each forest  $\mathbb{F}_j$  exists, this is true if the forward variance curve is bounded on finite intervals, in that case each tree is of order  $(T-t)^{j\alpha+1}$ . From the calculations that we made before on  $\mathbb{F}_k$  we have that

$$\tilde{\mathbb{F}}_k(u) = \beta_k(u) \theta^k I_t^{(k)}(T)$$



with  $\beta_k(u)$  a coefficient dependent on  $k$  and  $u$ . Define also  $\tilde{X}(u) = iux_t$ . Firstly we compute for  $l < j$

$$\begin{aligned}\tilde{\mathbb{F}}_l(u) \diamond \tilde{\mathbb{F}}_j(u) &= \theta^{l+j+2} \beta_l(u) \beta_j(u) \frac{\Gamma(l\alpha+1)\Gamma(j\alpha+1)}{\Gamma(l\alpha+\alpha+1)\Gamma(j\alpha+\alpha+1)} I_t^{(l+j+2)}(T) \\ \tilde{X}(u) \diamond \tilde{\mathbb{F}}_k(u) &= iu\rho\theta^k \frac{\Gamma(k\alpha-\alpha+1)}{\Gamma(k\alpha+1)} \beta_{k-1}(u)\end{aligned}$$

Using the recursion formula in **Definition 3.** with the correct modifications (substituting  $\mathbb{F}_j$  with  $\tilde{\mathbb{F}}_j(u)$  and  $x$  with  $\tilde{X}(u)$ ) we obtain a recursion formula to express the coefficients  $\beta_j(u)$  as

$$\begin{aligned}\beta_0(u) &= -\frac{u^2 + iu}{2} \\ \beta_k(u) &= \frac{1}{2} \sum_{l=0}^{k-2} \sum_{j=0}^{k-2} \mathbf{1}_{l+j=k-2} \beta_l(u) \beta_j(u) \frac{\Gamma(l\alpha+1)\Gamma(j\alpha+1)}{\Gamma(l\alpha+\alpha+1)\Gamma(j\alpha+\alpha+1)} \\ &\quad + iu\rho \frac{\Gamma(k\alpha-\alpha+1)}{\Gamma(k\alpha+1)} \beta_{k-1}(u)\end{aligned}$$

and now defining  $h(u, t)$  as the formal power series

$$h(u, t) = \sum_{j=0}^{\infty} \frac{\Gamma(\alpha j + 1)}{\Gamma(\alpha j + \alpha + 1)} \beta_j(u) \theta^j t^{(j+1)\alpha}$$

we have that for small times is converging, satisfy the fractional Riccati equation, the boundary condition and, moreover, thanks to what we noticed before

$$\phi_{rH}(u, T; t) = e^{\sum_{j=1}^{\infty} \tilde{\mathbb{F}}_j(u)} \phi(u, T; t) = \exp \left\{ iux_t + \int_t^T D^\alpha h(u, T-s) \xi_t(s) ds \right\}$$

which was exactly what we obtained in the previous sections (minus the drift term). Notice that the function  $\theta h(u, t)$  depends only on the quantity  $\theta t^\alpha$ , so if we do the following change of variable  $\tilde{t}^\alpha = \theta t^\alpha$  we can rewrite our fractional differential equation as

$$\begin{aligned}D^\alpha h(u, \tilde{t}) &= -\frac{u^2 + ui}{2} + iu\rho h(u, \tilde{t}) + \frac{1}{2} h(u, \tilde{t})^2 \\ &= \frac{1}{2} [h(u, \tilde{t}) - r_-] [h(u, \tilde{t}) - r_+]\end{aligned}$$

with  $r_\pm = -iu\rho \pm \sqrt{u^2 + iu - \rho^2 u^2}$ .

### 4.3.2 Long times expansion

If we define  $C := (r_+ - r_-)/2$ . Then this proposition holds:

**Proposition 10.** Let  $h_\infty(u, \tilde{t}) := r_-[1 - E_\alpha(-C\tilde{t}^\alpha)]$ . For  $u \in \mathbb{C}$  with  $\Re(u) \geq 0$  and  $\tilde{t} \in \mathbb{R}^+$ , if  $\tilde{t} \rightarrow \infty$  then  $h_\infty(u, \tilde{t})$  solves the fractional equation in **Theorem 3.** up to an error term of  $\mathcal{O}(|C\tilde{t}^\alpha|^{-2})$ .

To prove this proposition is sufficient to apply the last property shown in **section 2.7** and use the lemma. Notice also that the definition of  $C$  in the lemma is coherent with the definition of  $C$  in this section. Looking at **Proposition 4.** it raises a natural *ansatz* for  $h(u, \tilde{t})$  when  $\tilde{t} \rightarrow \infty$  and it is

$$h(u, \tilde{t}) = r_- \sum_{j=0}^{\infty} \gamma_j \frac{\tilde{t}^{-j\alpha}}{C^j \Gamma(1-j\alpha)}$$

for some coefficients  $(\gamma_j)_{j=0}^{\infty}$ . Using now the last property of Mittag-Leffler functions we, after changing the index, obtain

$$\frac{1}{r_-} D^\alpha h(u, \tilde{t}) = C \sum_{j=1}^{\infty} \gamma_{j-1} \frac{\tilde{t}^{-j\alpha}}{C^j \Gamma(1-j\alpha)}$$

then, assuming that our *ansatz* is the solution to the fractional differential equation we have also the following

$$\begin{aligned} \frac{1}{r_-} D^\alpha h(u, \tilde{t}) &= \frac{1}{r_-} \frac{1}{2} (h(u, \tilde{t}) - r_-)(h(u, \tilde{t}) - r_+) \\ &= \sum_{j=1}^{\infty} \gamma_j \frac{\tilde{t}^{-j\alpha}}{C^j \Gamma(1-j\alpha)} \left( -C + \frac{r_-}{2} \sum_{j=1}^{\infty} \gamma_j \frac{\tilde{t}^{-j\alpha}}{C^j \Gamma(1-j\alpha)} \right) \end{aligned}$$

using the identity principle for power series we obtain

$$\begin{aligned} \gamma_0 &= 1 \\ \gamma_1 &= -1 \\ \gamma_2 &= 1 + \frac{r_-}{2C} \frac{\Gamma(1-2\alpha)}{\Gamma(1-\alpha)^2} \\ &\vdots \\ \gamma_j &= -\gamma_{j-1} + \frac{r_-}{2C} \sum_{i=1}^{\infty} \sum_{l=1}^{\infty} \mathbf{1}_{i+l=j} \gamma_i \gamma_l \frac{\Gamma(1-j\alpha)}{\Gamma(1-i\alpha)\Gamma(1-l\alpha)} \end{aligned}$$

#### 4.3.3 Padé approximation

We define the rational approximation of  $h(u, \tilde{t})$  with  $m$  terms at the numerator and  $n$  terms at the denominator as

$$h^{(m,n)}(u, \tilde{t}) = \frac{\sum_{j=1}^m p_j \tilde{t}^{j\alpha}}{\sum_{j=1}^n q_j \tilde{t}^{j\alpha}}$$

such that

$$h^{(m,n)}(u, \tilde{t}) \sum_{j=1}^n q_j \tilde{t}^{j\alpha} - \sum_{j=1}^m p_j \tilde{t}^{j\alpha} = \mathcal{O}(\tilde{t}^{\alpha(m+n+1)})$$

Notice that for  $t \rightarrow \infty$ , thanks to the expansion for long times, we have that

$$h^{(m,n)}(u, \tilde{t}) \sim \frac{p_m \tilde{t}^{m\alpha}}{q_n \tilde{t}^{n\alpha}}$$

and we have seen that for long times it must be finite, so this can happen if and only if  $m = n$ . So the only admissible approximation of  $h$  are the ones of type  $h^{(m,m)}$ . As denoted in the paper we will choose  $m = 3$ , although there is no theoretical motivation for this choice, in practice happens to be both fast to compute and sufficiently accurate for our scope. Then, setting WLOG the constant at the denominator equal to 1:

$$h^{(3,3)}(u, \tilde{t}) = \frac{p_1 \tilde{t}^\alpha + p_2 \tilde{t}^{2\alpha} + p_3 \tilde{t}^{3\alpha}}{1 + q_1 \tilde{t}^\alpha + q_2 \tilde{t}^{2\alpha} + q_3 \tilde{t}^{3\alpha}}$$

thanks to the small time expansion for small  $x$  we have, for some coefficients  $b_1, b_2, b_3$ :

$$h(u, \tilde{t}) = b_1 \tilde{t}^\alpha + b_2 \tilde{t}^{2\alpha} + b_3 \tilde{t}^{3\alpha} + \mathcal{O}(\tilde{t}^{4\alpha})$$

meanwhile for long times and coefficients  $c_0, c_1, c_2$ :

$$h(u, \tilde{t}) = c_0 + \frac{c_1}{\tilde{t}^\alpha} + \frac{c_2}{\tilde{t}^{2\alpha}} + \mathcal{O}(\tilde{t}^{-3\alpha})$$

using the definition at the beginning of this section we obtain the following equations

$$\begin{aligned} p_1 &= b_1 \\ p_2 - p_1 q_1 &= b_2 \\ p_1 q_1^2 - p_1 q_2 - p_2 q_1 + p_3 &= b_3 \\ p_3 &= c_0 q_3 \\ p_2 q_3 - p_3 q_2 &= c_1 q_3^2 \\ p_1 q_3^2 - p_2 q_2 q_3 - p_3 q_1 q_3 + p_3 q_2^2 &= c_2 q_3^3 \end{aligned}$$

this linear system can be solved and the solution is

$$\begin{aligned}
p_1 &= b_1 \\
p_2 &= \frac{b_1^3 c_1 + b_1^2 c_0^2 + b_1 b_2 c_0 c_1 - b_1 b_3 c_0 c_2 + b_1 b_3 c_1^2 + b_2^2 c_0 c_2 - b_2^2 c_1^2 + b_2 c_0^3}{b_1^2 c_2 + 2b_1 c_0 c_1 + b_2 c_0 c_2 - b_2 c_1^2 + c_0^3} \\
p_3 &= c_0 q_3 \\
q_1 &= \frac{b_1^2 c_1 - b_1 b_2 c_2 + b_1 c_0^2 - b_2 c_0 c_1 - b_3 c_0 c_2 + b_3 c_1^2}{b_1^2 c_2 + 2b_1 c_0 c_1 + b_2 c_0 c_2 - b_2^2 c_1^2 + c_0^3} \\
q_2 &= \frac{b_1^2 c_0 - b_1 b_2 c_1 - b_1 b_3 c_2 + b_2^2 c_2 + b_2 c_0^2 - b_3 c_0 c_1}{b_1^2 c_2 + 2b_1 c_0 c_1 + b_2 c_0 c_2 - b_2^2 c_1^2 + c_0^3} \\
q_3 &= \frac{b_1^3 + 2b_1 b_2 c_0 + b_1 b_3 c_1 - b_2^2 c_1 + b_3 c_0^2}{b_1^2 c_2 + 2b_1 c_0 c_1 + b_2 c_0 c_2 - b_2^2 c_1^2 + c_0^3}
\end{aligned}$$

notice that  $I^{1-\alpha} h^{(3,3)}(u, 0) = 0$ .

## 5 Pricing

Using the facts that  $h$  is a solution of the fractional Riccati equation and that  $h^{(3,3)}$  approximates it well then we have:

$$\begin{aligned}
\phi_{rH}(u, T; 0) &= \exp \left\{ iux_0 + iu(r - q)T - \int_0^T \frac{u^2 + iu}{2} \xi_0(s) ds \right. \\
&\quad + \int_0^T iu\theta\rho h(u, T - s) \xi_0(s) ds \\
&\quad \left. + \int_0^T \frac{\theta^2}{2} h^2(u, T - s) \xi_0(s) ds \right\}
\end{aligned}$$

and then

$$\begin{aligned}
\hat{\phi}_{rH}(u, T; 0) &= \exp \left\{ iux_0 + iu(r - q)T - \int_0^T \frac{u^2 + iu}{2} \xi_0(s) ds \right. \\
&\quad + \int_0^T iu\theta\rho h^{(3,3)}(u, T - s) \xi_0(s) ds \\
&\quad \left. + \int_0^T \frac{\theta^2}{2} [h^{(3,3)}(u, T - s)]^2 \xi_0(s) ds \right\}
\end{aligned}$$

where  $\hat{\phi}_{rH}(u, T; 0) \approx \phi_{rH}(u, T; 0)$ . Now if we denote the volatility of a variance swap at time 0 with tenor  $T$  with  $\hat{\sigma}_0^T$  and remember the definition of the forward variance curve we have:

$$T(\hat{\sigma}_0^T)^2 = \int_0^T \xi_0(s) ds$$

where we have to notice also that  $\hat{\sigma}_0^T$  is the strike of a variance swap, which, for some tenors, can be observed in the market. The parametrization that we chose to use for the volatility of a variance swap is the Gompertz function, which is the following

$$\hat{\sigma}_0^T = z_1 e^{-z_2 e^{-z_3 T}}$$

where  $z_1, z_2$  and  $z_3$  are positive constants. The effects of the parameters are:  $z_1$  is the asymptote (i.e. the long time implied future volatility),  $z_2$  sets the displacement along the  $x$ -axis (i.e. time to maturity) and  $z_3$  sets the growth rate. We used the least squared method to fit the parameters and, on our dataset, we obtained:

$$z_1 = 0.2393444554 \quad z_2 = 0.2355916752 \quad z_3 = 0.1927188249$$

using these facts the formula becomes:

$$\begin{aligned} \hat{\phi}_{rH}(u, T; 0) = \exp \Big\{ iux_0 + iuT \Big[ r - q + (iu - 1) \frac{(\hat{\sigma}_0^T)^2}{2} \Big] \\ + iu\theta\rho \int_0^T h^{(3,3)}(u, T-s) \xi_0(s) ds \\ + \frac{\theta^2}{2} \int_0^T [h^{(3,3)}(u, T-s)]^2 \xi_0(s) ds \Big\} \end{aligned}$$

Now that we have a good approximation of the characteristic formula we can apply the Lewis's formula to evaluate the price of an european call option as

$$C_{0,K} = S_0 e^{-qT} - \frac{\sqrt{S_0 K} e^{-(r+q)T/2}}{\pi} \int_0^\infty \frac{\Re\{\phi_{rH}(u - i/2, T; 0)\}}{u^2 + 1/4} du$$

where the integral is performed numerically using standard integration techniques. Due to the intrinsic structure of the scripting language that we chose to use (Python) we have implemented this formula. However, we wish also to present a viable alternative in the next section.

## 5.1 Fourier transform technique for Vanilla Options

An alternative method has been developed by Carr and Madan to compute vanilla options prices leveraging the Fast Fourier Transform algorithm. As in the Heston model, we have a closed (but approximated) formula for the characteristic function. If we denote with  $f_{rH}$  the risk-neutral density function (conditional to time  $T$ ) then

$$\phi_{rH}(u, T; 0) = \int_{\mathbb{R}} e^{iux} f_{rH}(x) dx$$

so, in other words, the characteristic function is the Fourier transform of the transition density function. Consequently, if we denote with  $k$  the log of the

strike, the initial value of a call with strike  $K$  and tenor  $T$  is

$$C_{0,K} = \int_k^\infty e^{-rT} (e^s - e^k) f_{rH}(s) ds$$

and if  $k \rightarrow -\infty$ , or equivalently  $K \rightarrow 0$ , the value of a call with such a strike is equal to the discounted value of the spot at that time, hence the function is not in  $\mathcal{L}^2(\mathbb{R})$ . For the next considerations this property is needed, so we will dampen the call price with an exponential kernel. The modified price is defined as

$$c_{0,K} = e^{\beta k} C_{0,K}$$

with  $\beta > 0$ . The Fourier transform of  $c_{0,K}$  is

$$\begin{aligned} \psi(u, T; 0) &= \int_{\mathbb{R}} e^{iuk} \int_k^\infty e^{\beta k} e^{-rT} (e^s - e^k) f_{rH}(s) ds dk \\ &= \frac{e^{-rT} \phi_{rH}(u - (\beta + 1)i, T; 0)}{\beta^2 + \beta - u^2 + iu(2\beta + 1)} \end{aligned}$$

so the price of the call can be obtained using the inverse transform as

$$\begin{aligned} C_{0,K} &= \frac{e^{-\beta k}}{2\pi} \int_{\mathbb{R}} e^{-iuk} \psi(u, T; 0) du \\ &= \frac{e^{-\beta k}}{\pi} \int_0^\infty e^{-iuk} \psi(u, T; 0) du \\ &= \frac{e^{-\beta k}}{\pi} \int_0^\infty e^{-iuk} \frac{e^{-rT} \phi_{rH}(u - (\beta + 1)i, T; 0)}{\beta^2 + \beta - u^2 + iu(2\beta + 1)} du \end{aligned} \tag{12}$$

where in the second step we used the fact that  $C_{0,K} \in \mathbb{R}$  so the imaginary part of the integral must be odd and even in its real part. For the modified call value to be integrable in the positive log strike direction, and hence for it to be square-integrable as well, a sufficient condition is provided by  $\psi(0, T; 0)$  being finite, which is equivalent to  $\phi_{rH}(-(\beta + 1)i, T; 0)$  being finite and this happens if and only if

$$\mathbb{E}[S_T^{\beta+1}] < \infty$$

if we find a suitable  $\beta$  then we have that  $\psi \in \mathbb{R}^+$ , indeed  $\phi_{rH}$  is bounded by  $\mathbb{E}[S_T^{\beta+1}]$ , which is independent from  $u$  and then

$$|\psi(u, T; 0)|^2 \leq \frac{\text{cost.}}{u^4}$$

this also give us an estimation for the truncation error in the right tail, indeed

$$\int_a^\infty |\psi(u, T; 0)| du < \frac{\sqrt{\text{cost.}}}{a}$$

so we can choose a suitable  $a$  such that the truncation error in computing the transform is below a certain tolerance.

## 5.2 Numerical implementation

Let consider the first step in (4), if we truncate the integral till  $a$  and choose  $N$  and  $\Delta$  such that  $a = N\Delta$  we can discretize the integral, using the Simpson's rule, on a grid  $[0, \Delta, \dots, N\Delta]$  and obtain

$$C_{0,K} \approx \frac{e^{-\beta k}}{\pi} \sum_{j=1}^N e^{-i\Delta(j-1)k} \psi(\Delta(j-1), T; 0) \frac{\Delta}{3} [3 + (-1)^j - \delta_{j-1}]$$

where  $\delta_j$  is the Kronecker delta. Suppose now that we are interested mainly in at-the-money call values, so with  $k$  near 0. The FFT returns  $N$  values of  $k$  and we use a regular spacing size  $\lambda$ , so our grid for the strikes is  $[-b := -N\lambda/2, \dots, N\lambda/2 - \lambda] = [k_1, \dots, k_N]$  so the value at time 0 of a call option with strike  $K_v$  with  $k_v$  in the grid becomes

$$C_{0,K_v} \approx \frac{e^{-\beta k_v}}{\pi} \sum_{j=1}^N e^{-i\lambda\Delta(j-1)(v-1)} e^{ib\Delta(j-1)} \psi(\Delta(j-1), T; 0) \frac{\Delta}{3} [3 + (-1)^j - \delta_{j-1}]$$

and in order to use the FFT algorithm we need

$$\lambda\Delta = \frac{2\pi}{N}$$

for the strikes that are not in the grid we will use spline interpolation.

**NOTE:** Carr and Madan developed also a modification to price far OTM options, here we will not show the process, but only the final formula

$$C_{0,K_v} \approx \frac{\Delta/3\pi}{\sinh(-\beta k_v)} \sum_{j=1}^N e^{-i\lambda\Delta(j-1)(v-1)} e^{ib\Delta(j-1)} \gamma(\Delta j - \Delta) [3 + (-1)^j - \delta_{j-1}]$$

where

$$\begin{aligned} \zeta(u) &= e^{-rT} \left( \frac{1}{1+iu} - \frac{e^{rT}}{iv} - \frac{\phi_{rH}(u-i, T; 0)}{u^2 - iu} \right) \\ \gamma(u) &= \frac{\zeta(u-i\beta) - \zeta(u+i\beta)}{2} \end{aligned}$$

## 6 Calibration

The calibration is done in two steps. Firstly, we use a least square algorithm to fit the prices of the variance swaps to the ones observed in the market. In this way we fix  $z_1, z_2$  and  $z_3$ . After that, let  $\Xi := [H, \rho, \theta]^\top$  be our vector of parameters. We denote with  $\sigma^*(K_i, T_i)$  the market implied volatility for calls with strike  $K_i$  and maturity  $T_i$  and with  $\sigma(\Xi; K_i, T_i)$  the implied volatility for

calls under the rough Heston model with parameters  $\Xi$ . Given  $n$  call options we define:

$$r_i(\Xi) := \sigma(\Xi; K_i, T_i) - \sigma^*(K_i, T_i) \quad i = 1, \dots, n$$

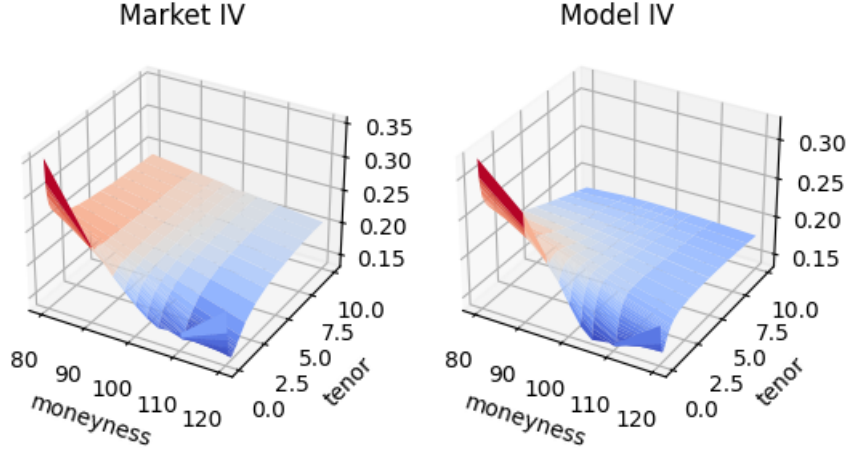
and the residual vector  $r(\Xi) = [r_1(\Xi), \dots, r_n(\Xi)]^\top$ . Then we have to remember that the parameters are subjected to certain constraints:  $H \in (0, 1/2)$ ,  $\theta > 0$  and  $\rho \in [-1, 1]$ . In **Theorem 3.** we used the additional hypothesis  $\rho \in (-1/\sqrt{2}, 1/\sqrt{2})$ , however, this is not strictly necessary and using the forests approach is possible to obtain the same result with  $\rho \in [-1, 1]$ . With this notation the calibration of the Heston model is an inverse problem in the nonlinear least square form as:

$$\min_{\substack{\Xi \\ H \in (0, 1/2) \\ \theta > 0 \\ \rho \in [-1, 1]}} \frac{1}{2} \|r(\Xi)\|^2$$

since we suppose to have  $n \gg 5$  (where 5 is the number of parameters that we have to determine) it is an overdetermined problem. To tackle this kind of problem we will use the Trust Region Reflective algorithm. For the interested reader see Branch, Coleman, Li in *A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems*.

## 6.1 Numerical results

Here we report the implied volatility surface calibrated to all the strikes and tenors together:



where the parameters are:

$$\theta = 0.2817 \quad H = 0.0010 \quad \rho = -0.6995$$



the mean relative percentage error obtained is 6.4480% and it took around 3 mins on a standard laptop. As we can see we were able to obtain pretty similar surface especially for shorter tenors, meanwhile for longer tenors seems that we are underestimating the implied volatility. It is interesting to notice that we have obtained a similar  $\rho$  for both the Heston and the rough Heston model.

## 7 Simulation

In this section we will explain the HQE scheme proposed by Gatheral in *Efficient simulation of affine forward variance models* to simulate price paths under the rough Heston model. Firstly, we will rewrite the rough Heston model in the forward variance form and under the log-spot dynamics as:

$$\begin{cases} dx_t = \left( r - q - \frac{v_t}{2} \right) dt + \sqrt{v_t} d\tilde{W}_t \\ d\xi_t(u) = \frac{\theta}{\Gamma(H + \frac{1}{2})} \frac{\sqrt{v_t}}{(u-t)^{\frac{1}{2}-H}} dW_t =: \kappa(u-t) \sqrt{v_t} dW_t \end{cases}$$

If we discretize the variance process with timestep  $\Delta = T/N$  (where  $N$  is the number of steps) we obtain:

$$v_{n\Delta} = \xi_0(n\Delta) + \sum_{k=1}^n \int_{(k-1)\Delta}^{k\Delta} \kappa(n\Delta - s) \sqrt{v_s} dW_s$$

we can now define the  $F_{n-1}$ -adapted part as

$$\hat{\xi}_n := \xi_0(n\Delta) + \sum_{k=1}^{n-1} \int_{(k-1)\Delta}^{k\Delta} \kappa(n\Delta - s) \sqrt{v_s} dW_s$$

and the martingale increment as

$$u_n = \int_{(n-1)\Delta}^{n\Delta} \kappa(n\Delta - s) \sqrt{v_s} dW_s$$

so

$$v_{n\Delta} = \hat{\xi}_n + u_n$$

moreover let's denote also for  $i, j \geq 0$

$$\mathcal{K}_i = \int_0^\Delta \kappa(s + i\Delta) ds \quad \text{and} \quad \mathcal{K}_{i,j} = \int_0^\Delta \kappa(s + i\Delta) \kappa(s + j\Delta) ds$$

and, doing the calculations, we obtain

$$\begin{aligned} \mathcal{K}_i &= \frac{\theta}{\alpha \Gamma(\alpha)} \Delta^\alpha [(i+1)^\alpha - i^\alpha] \\ \mathcal{K}_{i,i} &= \frac{\theta^2}{2H [\Gamma(\alpha)]^2} \Delta^{2H} [(i+1)^{2H} - i^{2H}] \end{aligned}$$

to conclude this introductory part we will state the following result, which can be obtained using the Itô's isometry:

**Lemma 11.** *Assuming that the forward variance curve is twice differentiable then*

$$\text{Var}[u_n|F_{n-1}] = \frac{\mathcal{K}_{0,0}}{2H+1} [\hat{\xi}_n + 2Hv_{(n-1)\Delta}] + \mathcal{O}(\Delta^{2+2H}) =: \bar{v}_n \mathcal{K}_{0,0} + \mathcal{O}(\Delta^{2+2H})$$

the last definition that we need is

$$\chi_n = \int_{(n-1)\Delta}^{n\Delta} \sqrt{v_s} dW_s$$

## 7.1 The HQE scheme

Using **Lemma 6.** we have that for  $s \in ((n-1)\Delta, n\Delta]$  it is true that  $v_s \approx \bar{v}_n$ , from there we get the corresponding approximations

$$\begin{aligned} \text{Var}[\hat{\xi}_{n+1}|F_{n-1}] &\approx \bar{v}_n \mathcal{K}_{1,1} \\ \text{Var}[\chi_n|F_{n-1}] &\approx \bar{v}_n \Delta \\ \text{Cov}[u_n, \hat{\xi}_{n+1}|F_{n-1}] &\approx \bar{v}_n \mathcal{K}_{0,1} \\ \text{Cov}[u_n, \chi_n|F_{n-1}] &\approx \bar{v}_n \mathcal{K}_0 \\ \text{Cov}[\chi_n, \hat{\xi}_{n+1}|F_{n-1}] &\approx \bar{v}_n \mathcal{K}_1 \end{aligned}$$

then the correlation matrix takes this form

$$\begin{pmatrix} 1 & \frac{\mathcal{K}_0}{\sqrt{\Delta}\sqrt{\mathcal{K}_{0,0}}} & \frac{\mathcal{K}_{0,1}}{\sqrt{\mathcal{K}_{1,1}}\sqrt{\mathcal{K}_{0,0}}} \\ \frac{\mathcal{K}_0}{\sqrt{\Delta}\sqrt{\mathcal{K}_{0,0}}} & 1 & \frac{\mathcal{K}_1}{\sqrt{\Delta}\sqrt{\mathcal{K}_{1,1}}} \\ \frac{\mathcal{K}_{0,1}}{\sqrt{\mathcal{K}_{1,1}}\sqrt{\mathcal{K}_{0,0}}} & \frac{\mathcal{K}_1}{\sqrt{\Delta}\sqrt{\mathcal{K}_{1,1}}} & 1 \end{pmatrix}$$

where all the entries are independent of the time step  $n$  and are all functions of only  $H$ . Moreover we will leverage the following approximations for a sufficiently small  $\Delta$

$$\mathcal{K}_{0,1} \approx \frac{\mathcal{K}_1 \mathcal{K}_0}{\Delta} \quad \text{and} \quad \mathcal{K}_{1,1} \approx \frac{\mathcal{K}_1^2}{\Delta}$$

with that our correlation matrix becomes

$$R = \begin{pmatrix} 1 & \frac{\sqrt{2H}}{H+1/2} & \frac{\sqrt{2H}}{H+1/2} \\ \frac{\sqrt{2H}}{H+1/2} & 1 & 1 \\ \frac{\sqrt{2H}}{H+1/2} & 1 & 1 \end{pmatrix}$$

### 7.1.1 The Andersen Quadratic Exponential scheme

Andersen developed the Quadratic Exponential scheme to integrate the Heston model which matches the means and the variances. Moreover this scheme also guarantees that  $v_{\Delta n}$  stays positive in every point. We need the following definitions:

$$\begin{aligned}\psi_n &= \frac{\text{Var}[u_n|F_{n-1}]}{\hat{\xi}_n^2} \\ \beta_n^2 &= \frac{2}{\psi_n} - 1 + \sqrt{\frac{2}{\psi_n}} \sqrt{\frac{2}{\psi_n} - 1} \\ Z_n &\sim \mathcal{N}(0, 1) \\ U_n &\sim \mathcal{U}(0, 1) \\ p_n &= \frac{2}{1 + \psi_n} \\ \gamma_n &= \frac{1}{2} \hat{\xi}_n (1 + \psi_n)\end{aligned}$$

Then the algorithm works in this way:

---

#### Algorithm 4 QE Scheme

---

- 1: Evaluate  $\psi_n$ .
- 2: **if**  $\psi_n \geq 3/2$  **then**
- 3:     Simulate  $v_n$  as

$$v_n = -\mathbf{1}_{U_n < p_n} \gamma_n \log \frac{U_n}{p_n}$$

- 4: **else**
- 5:     Simulate  $v_n$  as

$$v_n = \frac{\hat{\xi}_n}{1 + \beta_n^2} (\beta_n + Z_n)^2$$

- 6: **end if**
- 

here at each step we simulate only one random variable, Gatheral developed an extension of this for the rough Heston model.

### 7.1.2 Hybrid simulation step

Linear regression gives

$$u_n \approx \frac{\mathcal{K}_0}{\Delta} \chi_n + \varepsilon_n$$

where the three quantities on the RHS are uncorrelated. The next lemma guarantees the positivity of the variance process and that the covariance matrix presented in the previous session is correctly matched.

**Lemma 12.** Let  $\frac{\mathcal{K}_0}{\Delta}\tilde{\chi}_n = \frac{\mathcal{K}_0}{\Delta}\chi_n + \frac{\hat{\xi}_n}{2}$  and  $\tilde{\varepsilon}_n = \varepsilon + \frac{\hat{\xi}_n}{2}$  be generated using the QE scheme presented in the previous section with the following conditional mean and variances:

$$\begin{aligned}\mathbb{E}\left[\frac{\mathcal{K}_0}{\Delta}\tilde{\chi}_n|F_{n-1}\right] &= \frac{\hat{\xi}_n}{2}; \quad \mathbb{E}[\tilde{\varepsilon}_n|F_{n-1}] = \frac{\hat{\xi}_n}{2} \\ \text{Var}[\tilde{\chi}_n|F_{n-1}] &= \bar{v}_n\Delta; \quad \text{Var}[\tilde{\varepsilon}_n|F_{n-1}] = \bar{v}_n\left(\mathcal{K}_{0,0} - \frac{\mathcal{K}_0^2}{\Delta}\right)\end{aligned}$$

Then  $v_{n\Delta} = \frac{\mathcal{K}_0}{\Delta}\chi_n + \varepsilon_n + \hat{\xi}_n \geq 0$ . Moreover, with  $u_n = \frac{\mathcal{K}_0}{\Delta}\chi_n + \varepsilon_n$

$$\text{Var}[u_n|F_{n-1}] = \bar{v}_n\mathcal{K}_{0,0} \quad \text{and} \quad \text{Cov}[u_n, \chi_n|F_{n-1}] = \bar{v}_n\mathcal{K}_0$$

Now we have everything we need to present the HQE scheme, but before doing that we need to recognize that there is nothing which guarantees that  $\hat{\xi}_n$  is always positive. Indeed, we have to impose this additional condition by choosing an  $\epsilon > 0$  which will be the lower bound of the variance process.

---

**Algorithm 5** HQE Scheme

---

1: Given  $\chi_k$ , for  $k < n$ , with  $\epsilon$  very small, compute

$$\hat{\xi}_n = \max\left[\epsilon, \xi_0(n\Delta) + \sum_{k=1}^{n-1} \sqrt{\frac{\mathcal{K}_{n-k+2, n-k+2}}{\Delta}} \chi_k\right]$$

2: Simulate  $\tilde{\chi}_n$  and  $\tilde{\varepsilon}_n$  accordingly to **Lemma 7**.

3: Compute  $v_{n\Delta} = \frac{\mathcal{K}_0}{\Delta}\tilde{\chi}_n + \tilde{\varepsilon}_n$ .

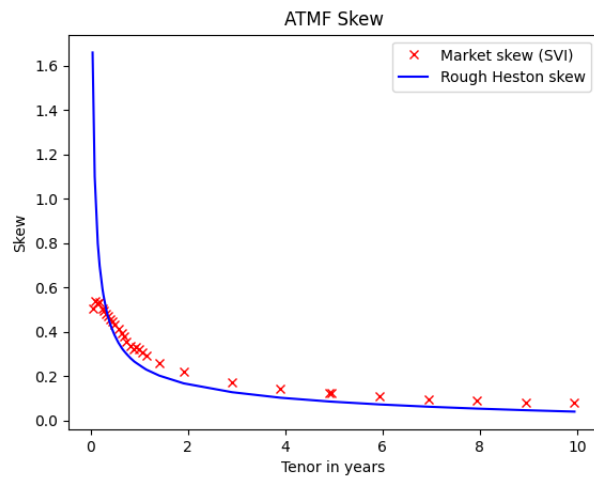
4: Update the log-price, with  $\tilde{v}_n = \frac{v_{n\Delta} + v_{(n+1)\Delta}}{2}$  and  $Z_n^\perp \sim \mathcal{N}(0, 1)$  as

$$X_{n\Delta} = X_{(n-1)\Delta} + \left(r - q - \frac{\tilde{v}_n}{2}\right)\Delta + \sqrt{1 - \rho^2}\sqrt{\Delta\tilde{v}_n}Z_n^\perp + \rho\chi_n$$


---

## 8 Is Heston's ATMF skew problem solved?

In the previous chapter we have seen that the Heston model does not fit well the at-the-money forward skew, mainly because the ATMF decays as  $1/T$  for  $T \rightarrow \infty$ . If we repeat the same process as in that paragraph with the calibrated rough Heston we will obtain:



a much better fit! This is due to the fact that we are using a power-law kernel for the fractional Brownian motion, and for  $T \rightarrow \infty$  behaves like  $T^{-\alpha} \approx T^{-0.5}$ .

## References

1. Gatheral, Jaisson, Rosenbaum. Volatility is rough. ArXiv, 2014.
2. Bouchaud, The endogenous dynamics of markets: price impact and feedback loops. ArXiv, 2010.
3. Brunnermeier, Pedersen. Market Liquidity and Funding Liquidity. The Review of Financial Studies, 2009.
4. El Euch, Fukasawa, Rosenbaum. The microstructural foundations of leverage effect and rough volatility. ArXiv, 2016.
5. Jaisson, Rosenbaum. Rough fractional diffusions as scaling limits of nearly unstable heavy tailed Hawkes processes. ArXiv, 2015.
6. El Euch, Gatheral, Rosenbaum. Roughening Heston. Risk, 2019.
7. El Euch, Gatheral, Rosenbaum. Perfect hedging in rough Heston models. ArXiv, 2017.
8. El Euch, Rosenbaum. The characteristic function of rough Heston models. ArXiv, 2016.
9. Gatheral, Radoicic. Rational approximation of the rough Heston solution. International Journal of Theoretical and Applied Finance, 2019.
10. Alòs, Gatheral, Radoicic. Exponentiation of conditional expectations under stochastic volatility. Quantitative Finance, 2020.
11. Branch, Coleman, Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. SIAM Journal of scientific computing, 1999.
12. Gatheral. Efficient simulation of affine forward variance models. Risk, 2022.
13. Leif Andersen. Simple and efficient simulation of the Heston stochastic volatility model. The Journal of Computational Finance, 2008.
14. Bennedsen, Lunde and Pakkanen. Hybrid scheme for Brownian semistationary processes. ArXiv, 2017.
15. Lewis. A simple option formula for general jump-diffusion and other exponential Lévy processes. OptionCity.net, 2001.
16. Carr, Madan. Option valuation using the fast Fourier transform. Journal of computational finance, 2001.

## Part IV

# Bayesian Inference

The main goal of this thesis is to leverage a Bayesian framework in order to infer from market data which model other market participants are employing to price options. We will show only the concept using four different models, but it can be extended to include any arbitrary number of models. In the first part of the chapter we will briefly introduce the rough Bergomi model and the Quintic Ornstein Uhlenbeck model, which are explained in depth in the thesis of Alessandro Nodari, *Bayesian framework to quantify rough volatility calibration risk*.

## 1 Additional models

The two following models are rough stochastic volatility models. Regarding the rough Bergomi model, it has the same property of the rough Heston model to represent well the forward at-the-money skew. The newer Quintic Ornstein Uhlenbeck model is a try to tackle the problem of the joint calibration. This is basically the idea to calibrate simultaneously the options on SPX, the futures on VIX and the options on VIX. Unfortunately, they both rely on Monte Carlo simulation for their calibration, since there is no closed or semiclosed formulae for pricing european vanilla options.

### 1.1 Rough Bergomi model

If we define  $C_H$  as

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

the solutions to the dynamic (under the risk-free measure) for the stock price process and for the volatility process of the rough Bergomi model are

$$\begin{cases} S_t = S_0 \exp \left\{ (r - q)t - \frac{1}{2} \int_0^t v_u du + \int_0^t \sqrt{v_u} d\tilde{W}_u \right\} \\ v_t = \xi_0(t) \exp \left\{ 2\nu C_H \int_0^t (t - u)^{H - \frac{1}{2}} dW_u - \frac{\nu^2 C_H^2}{H} t^{2H} \right\} \end{cases}$$

if  $t_0 = 0$  is the starting time. Where:

- $r$  is the risk-free rate;
- $q$  is the yield of the underlying;
- $W$  and  $\tilde{W}$  are correlated Brownian motions with  $d\langle \tilde{W}, W \rangle_t = \rho dt$ ;
- $2\nu$  is the volatility of the volatility;

- $H \in (0, 1/2)$  is the Hurst exponent of the fractional Brownian motion;
- $\xi_0(\cdot)$  is the forward variance curve as in the rough Heston model.

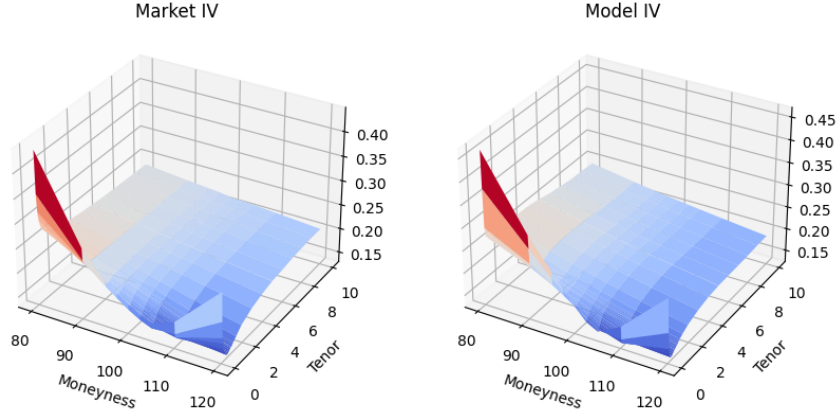
The three parameters  $(H, \rho, \nu)$  parameters have these effects on the implied volatility surface:  $H$  controls the decay of the term structure of volatility skew for very short expirations whereas the product  $\rho\nu$  sets the level of the ATM skew for longer tenors.

### 1.1.1 Calibration and Numerical Results

As we said before, we do not have a closed formula for the characteristic function, so we have to rely on Monte Carlo simulation. In order to perform the simulation a numerical integration scheme is needed. The scheme implemented by Nodari is the one adapted from McCrickerd and Pakkanen in *Turbocharging Monte Carlo pricing for the rough Bergomi model*. This is basically a forward Euler scheme paired with an hybrid scheme to simulate the Volterra process

$$\mathcal{V}_t = \int_0^t (t-u)^{H-\frac{1}{2}} dW_u$$

for more information please refer to his work. Here we report the implied volatility surface he obtained calibrated to all the strikes and tenors together:



where the parameters are:

$$\nu = 1.8906 \quad H = 0.0856 \quad \rho = -0.8978$$

the mean relative percentage error obtained is 3.1008% and it took around 87 mins on a standard laptop, but the algorithm is highly parallelizable.



## 1.2 Quintic Ornstein Uhlenbeck model

The Quintic Ornstein Uhlenbeck volatility model, introduced by Jaber, Illand, and Li, is a stochastic volatility model where the volatility process is defined as a polynomial of degree five of a single Ornstein Uhlenbeck process which has a fast mean reversion and a large volatility of volatility. Under the risk-free measure the dynamic is the following:

$$\begin{cases} dS_t &= (r - q)dt + v_t S_t d\tilde{W}_t \\ v_t &= \sqrt{\xi_0(t)} \frac{p(X_t)}{\sqrt{\mathbb{E}[p(X_t)^2]}} \\ dX_t &= (H - 1/2)\varepsilon^{-1} X_t dt + \varepsilon^{H-1/2} dW_t \end{cases}$$

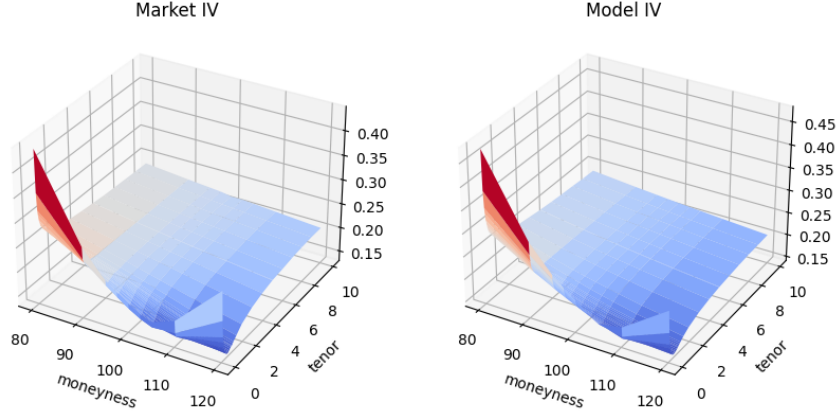
where  $X_0 = 0$ . The parameters are the following:

- $r$  is the risk-free rate;
- $q$  is the yield of the underlying;
- $W$  and  $\tilde{W}$  are correlated Brownian motions with  $d\langle \tilde{W}, W \rangle_t = \rho dt$ ;
- $2\nu$  is the volatility of the volatility;
- $H \in (0, 1/2)$  plays a similar role to the Hurst exponent of the fractional Brownian motion;
- $\varepsilon$  controls the mean reversion speed (with  $H$ ) and the volatility of the volatility;
- $\xi_0(\cdot)$  is the forward variance curve as in the rough Heston model;
- $p(x) := \alpha_0 + \alpha_1 x + \alpha_3 x^3 + \alpha_5 x^5$  is a fifth-grade polynomial with non-negative coefficients.

The choice of a polynomial of degree five allows to reproduce the upward slope of the VIX skew, while restricting the coefficients  $\alpha$  to be non-negative allows the sign of the ATM skew to be the same as  $\rho$ . For  $\varepsilon$  close to zero we have a fast mean-reversion regime and high volatility of volatility, which are two desirable properties to match empirical data.

### 1.2.1 Calibration and Numerical Results

Calibration has been performed in a similar way as in the rough Bergomi model. Here we report the implied volatility surface he obtained calibrated to all the strikes and tenors together:



where the parameters are:

$$\begin{aligned} \rho &= -0.93924 & H &= 0.106789 & \varepsilon &= 0.033779 \\ \alpha_0 &= 0.94018 & \alpha_1 &= 0.26251 & \alpha_3 &= 0.064811 & \alpha_5 &= 0.163232 \end{aligned}$$

the mean relative percentage error obtained is 3.5868% and it took around 52 mins on a standard laptop, but the algorithm is highly parallelizable. Moreover, in Nodari's work, you can also see the result obtained in the join-calibration with also the VIX futures and VIX options.

## 2 Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) methods are suitable for inferring posterior distributions in cases where the likelihood function is intractable or costly to evaluate and we have a simulator  $f(\cdot)$  to generate synthetic data, which, in our case, will be the stochastic volatility models introduced in the previous sections. The most vanilla ABC algorithm is an acceptance-rejection method. Given: a set of observations  $Y = \{y_j\}_{j=1,\dots,n}$ , a statistics  $s$ , a distance  $d$ , a tolerance threshold  $\varepsilon > 0$  and  $i = 0$ ; then the algorithm works as follow:

---

### Algorithm 6 Vanilla ABC

---

- 1: **while**  $i < N$  **do**
  - 2:   Sample a vector of parameters  $\theta$  from a prior distribution  $\pi(\cdot)$ .
  - 3:   Generate synthetic data  $Z = \{z_j\}_{j=1,\dots,n}$  using  $f(\theta)$ .
  - 4:   **while**  $d(s(Y), s(Z)) > \varepsilon$  **do**
  - 5:     Repeat step 2. and 3.
  - 6:   **end while**
  - 7:    $\theta_i = \theta$ ,  $i = i + 1$ .
  - 8: **end while**
  - 9: **return**  $\theta_1, \dots, \theta_N$
-

it can be proven that if  $s$  is a sufficient statistics then, for  $\varepsilon \rightarrow 0$ , the approximated posterior converges to the true posterior, otherwise it is not true and the limiting distribution is at best  $\pi(\theta|s(Y))$ . As we have seen the core idea of ABC is to bypass calculation of the likelihood by using simulations. However, we have to carefully choose the summary statistics  $s$ , Nadjahi et al. proposed the Sliced Wasserstein which will be expanded in the next subsection. Notice that in the practical implementation we have chosen to use the PyMC library which leverage the Sequential Monte Carlo ABC. SMC ABC iteratively morphs the prior into a posterior by propagating the sampled parameters through a series of proposal distributions, weighting the accepted parameters and allowing to sample from distributions with multiple peaks.

## 2.1 Sliced Wasserstein distance

The use of a statistics  $s$  may lead to a loss of information, in order to avoid that, Bernton et al. proposed to operate on the full data by using low-variance Wasserstein distances in terms of empirical distributions of observed and synthetic data. These express distance via an optimal transport problem of minimizing, with respect to an underlying distance metric, the cost of transforming a given probability measure into another one. Consider  $A \subset \mathbb{R}^n$ ,  $\mathcal{P}(A)$  as the set of probability measure on  $A$  and  $p \geq 1$ . Then we can define the space

$$\mathcal{P}_p(A) = \left\{ \mu \in \mathcal{P}(A) : \int_A \|y - y_0\|^p d\mu(y) < \infty \right\}$$

for some  $y_0 \in A$ . The  $p$ -Wasserstein distance between  $\mu, \nu \in \mathcal{P}_p(A)$  is then

$$W_p^p(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{A \times A} \|x - y\|^p d\gamma(x, y)$$

where  $\Gamma(x, y)$  is the set of probability measure on  $A \times A$  verifying that the marginal distributions are  $\mu$  and  $\nu$  respectively. As the solution of the optimal transport problem may be computationally challenging for high-dimensional problems, Nadjahi et al. suggested to project multi-dimensional distributions to one-dimensional ones via linear projections and then averaging 1D Wasserstein distances, which can be efficiently calculated by sorting, across various projections via a Monte-Carlo integral. Let  $\mathbb{S}^{d-1}$  be the  $d$ -dimensional unit sphere,  $u \in \mathbb{S}^{d-1}$ .  $u^*$  the linear form associated with  $u$ , such that  $\forall a \in A$ ,  $u^*(a) = \langle u, a \rangle$  where the inner product is the Euclidean one. The Sliced Wasserstein distance is then

$$SW_p^p(\mu, \nu) = \int_{\mathbb{S}^{d-1}} W_p^p(u_{\#}^* \mu, u_{\#}^* \nu) d\sigma(u)$$

where  $\sigma$  is the uniform distribution on the unit sphere and  $u_{\#}^* \mu$  denote the push-forward measure of  $\mu$  by  $u^*$ . This is still a distance on  $\mathcal{P}_p(A)$  and has significant lower computation requirement than the Wasserstein distance. Under some mild assumptions, they have also proved that the limiting posterior converges to the true posterior and also that if the number of samples tends to infinity then the

Sliced Wasserstein distance between two empirical distributions converges to the Sliced Wasserstein distance between the two real distributions from which the observations are drawn.

### 3 Infer the models

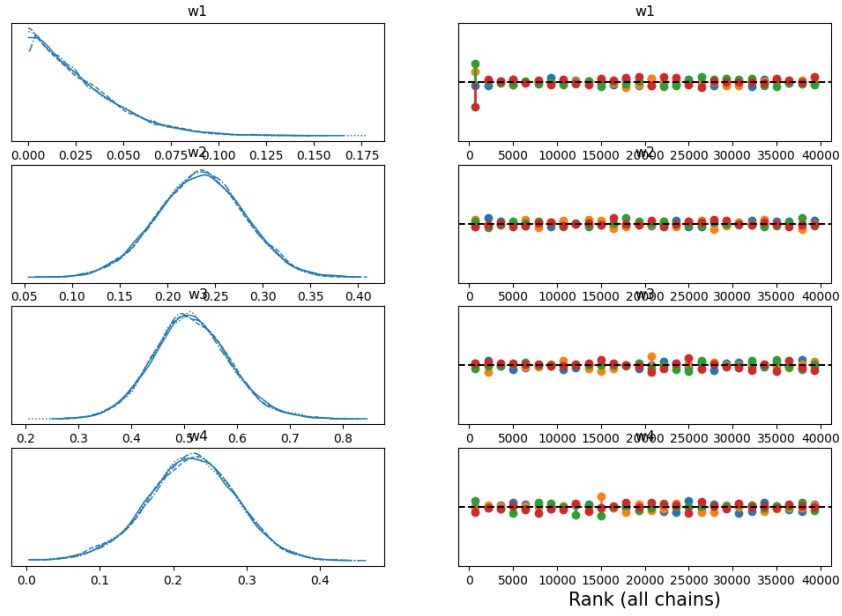
To infer the models used by other market participants we will proceed in the following way: firstly calibrate the models to market data, then proceed to find a convex combination of them using SMC ABC method showed in the previous chapter. Our set of models consists in four models: Heston, rough Heston, rough Bergomi and Quintic OU. Each one of them will have a weight, respectively  $w_1, w_2, w_3$  and  $w_4$ . We start with uninformative priors, because we do not have any knowledge about which model is prevailing. In order to ensure that the combination is convex we will use:

- $w_1 \sim \mathcal{U}(0, 1)$ ;
- $w_2 \sim \mathcal{U}(0, 1 - w_1)$ ;
- $w_3 \sim \mathcal{U}(0, 1 - w_1 - w_2)$ ;
- $w_4 = 1 - w_1 - w_2 - w_3$ .

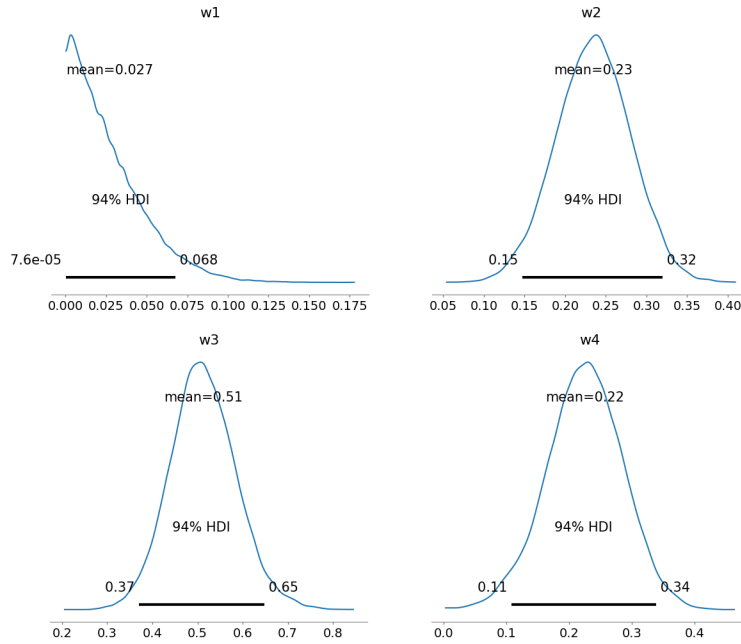
the observations will be the points of the implied volatility surface observed in the market. In this way we will retrieve the distributions for the weights and from there we can assess which models are mostly used. This can be expanded to any arbitrary number of models and, apart from calibrating the models itself, it is quite fast.

#### 3.1 Numerical results

We used 4 chains with 10000 samples each and  $\varepsilon = 0.1$ . The first figure represents on the left the distribution obtained from all the 4 chains superimposed, while in the right part we have a graphical representation of the rank of each chain: the vertical lines indicate deviation from the ideal expected value, which is represented with a black dashed line, if it is above we have more samples than expected and vice versa if it is below. We obtained the following results:



and the following posteriors



as we can see the market data are implying a rough Bergomi model, the mean of  $w_3$  is approximately 51%, meanwhile the mean of the other two rough models

is around 23%. The Heston model, even if it has not obtained a bad error in the calibration, seems to be not used. As a further empirical test, we also permuted the models and performed again the algorithm, but we did not obtain different results: rBergomi seems the one that can better explain market data, followed by rHeston and Quintic.

## 3.2 Calibration using ABC

Another possible application of this Bayesian framework is to assess calibration risk of the models. This is the what Nodari developed in his work. Whenever we calibrate a model we obtain only a point estimate of the parameters and, in most of the cases, we do not have much information about the geometry of the objective function. So it is possible that the parameters obtained are really unstable and this will result in a worse hedging of the claims. In this way we will not obtain a point estimate for the parameters, but a probability distribution, so we can quantify the errors, calculate the confidence intervals and much more. We will present here only the application of this process to the Heston model, please refer to his work for more in depth explanations.

### 3.2.1 ABC Heston calibration

The Heston model has 5 parameters which are not independent, in the sense that different parameters, under certain conditions, may lead to the same implied volatility surface. This is a problem with our framework, because with completely uninformative priors we will have a slow convergence. To avoid this situation the calibration will be divided into two parts. In the first one calibrate  $\kappa, \eta$  and  $\sigma_0$  using the theoretical price of a variance swap under Heston compared to the prices of variance swaps observed in the market. In the second one use the obtained distributions of  $\kappa, \eta$  and  $\sigma_0$  as priors, while using uninformative priors for  $\theta$  and  $\rho$ . The fair value for a variance swap with tenor  $T$  in Heston is:

$$\sigma_T = \sqrt{\eta + (\sigma_0 - \eta) \frac{1 - e^{-\kappa T}}{\kappa T}}$$

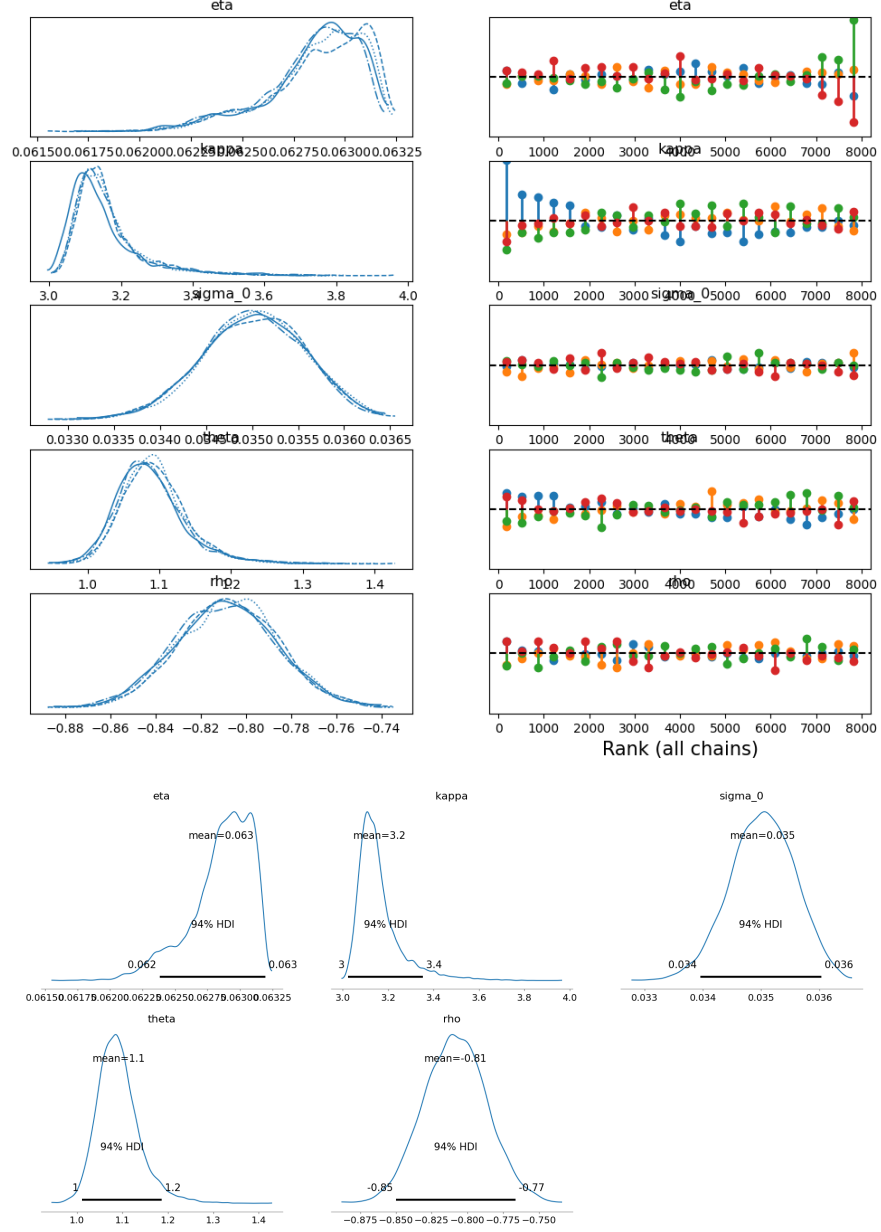
the priors used for the first part were:

- $\eta \sim \mathcal{U}(0, 2)$ ;
- $\kappa \sim \mathcal{U}(1, 7)$ ;
- $\sigma_0 \sim \mathcal{U}(0, 2)$ ;

meanwhile for the second part he used:

- $\eta, \kappa$  and  $\sigma_0$  discrete distribution obtained in the previous step;
- $\theta \sim \mathcal{U}(0.5, 3)$ ;
- $\rho \sim \mathcal{U}(-1, 0)$ ;

and obtained:



fixing the MAP value for each parameter he obtained a mean relative percentage error of 4.5024%. He also developed an indicator to assess the calibration risk, which is the absolute value of the difference between the errors weighted with the posterior densities and the error obtained with the deterministic calibration, the bigger it is this difference the bigger it is the calibration risk. As a

further extension we could join the two approaches and see the ensemble as an hierarchical model, this will require some time to run, but it will allow to create a model of models to investigate in the changes of the market.



## References

1. Nodari. Bayesian framework to quantify rough volatility calibration risk. Master's thesis, 2023.
2. Piiroinen, Roininen, Schoden and Simon. Asset Price Bubbles: An Option-based Indicator. ArXiv, 2018.
3. Jaber, Illand, and Li. The quintic Ornstein-Uhlenbeck volatility model that jointly calibrates SPX & VIX smiles. ArXiv, 2023.
4. Nadjahi, De Bortoli, Durmus, Badeau, Simsekli. Approximate Bayesian computation with the sliced-Wasserstein distance. ArXiv, 2020.
5. Bernton, Jacob, Gerber, Robert. Approximate Bayesian computation with the Wasserstein distance. ArXiv, 2019.

# Libraries

## Utils

### ImpliedDrift.py

```
1 import numpy as np
2 import pandas as pd
3 from scipy.interpolate import CubicSpline
4
5 dates = np.array(["23-01-23", "24-01-23", "25-01-23", "26-01-23",
6 "27-01-23", "30-01-23", "06-02-23", "13-02-23", "21-02-23"])
7 data = pd.read_csv("ratesOIS.csv")
8 tenor = np.array(data.TENOR)
9 Forw = pd.read_csv("forw.csv")
10 spot = np.array(pd.read_csv("spot.csv").Spot)
11 TENOR = pd.read_csv("tenor.csv")
12
13
14 def r(x, index = 0):
15     rates = np.array(data[dates[index]])/100
16     cs = CubicSpline(tenor, rates)
17     return cs(x)
18
19 def drift(x, index = 0):
20     S0 = spot[index]
21     F = np.array(Forw[dates[index]]).flatten()
22     Tenor = np.array(TENOR[dates[index]]).flatten()
23     d = -np.log(S0/F).flatten()/Tenor
24     cs = CubicSpline(Tenor, d)
25     return cs(x)
26
27 def q(x, index = 0):
28     return r(x, index) - drift(x, index)
```

### BlackScholes.py

```
1 import numpy as np
2 from scipy.stats import norm
3
4 def BSCall(S0, K, T, r, q, sigma):
5
6     # Price of a call under Black&Scholes
7
8     # S0: spot price
9     # K: strike
10    # T: years to expiration
11    # r: risk free rate (1 = 100%)
12    # q: annual yield
13    # sigma: volatility (1 = 100%)
14
15
16    sig = sigma*np.sqrt(T)
17    d1 = (np.log(S0/K) + (r-q)*T)/sig + sig/2.
```

```

18     d2 = d1 - sig
19
20     return S0*norm.cdf(d1) - K*np.exp(-(r-q)*T)*norm.cdf(d2)
21
22 def BSPut(S0, K, T, r, q, sigma):
23
24     # Price of a put under Black&Scholes
25
26     # S0: spot price
27     # K: strike
28     # T: years to expiration
29     # r: risk free rate (1 = 100%)
30     # q: annual yield
31     # sigma: volatility (1 = 100%)
32
33     return BSCall(S0, K, T, r, q, sigma) + K*np.exp(-(r-q)*T) - S0
34
35 def BSImpliedVol(S0, K, T, r, q, P, Option_type = 1, toll = 1e-10):
36
37     # Calculate implied volatility from prices using bisection
38
39     # NOTE: All the parameters can be np.array(), except for P that
40     # MUST be a np.array().
41
42     # S0: spot price
43     # K: strike
44     # T: years to expiration
45     # r: risk free rate (1 = 100%)
46     # q: annual yield
47     # P: prices
48     # Option_type: 1 for calls, 0 for puts
49     # toll: error in norm 1
50
51     if Option_type:
52         BSFormula = np.vectorize(BSCall)
53     else:
54         BSFormula = np.vectorize(BSPut)
55
56     N = P.shape[0]
57     sigma_low = 1e-10*np.ones(N)
58     sigma_high = 10*np.ones(N)
59
60     P_low = BSFormula(S0, K, T, r, q, sigma_low)
61     P_high = BSFormula(S0, K, T, r, q, sigma_high)
62     sigma = (sigma_low + sigma_high)/2.
63
64     while np.sum(P_high - P_low) > toll:
65         sigma = (sigma_low + sigma_high)/2.
66         P_mean = BSFormula(S0, K, T, r, q, sigma)
67         P_low += (P_mean < P)*(P_mean - P_low)
68         sigma_low += (P_mean < P)*(sigma - sigma_low)
69         P_high += (P_mean >= P)*(P_mean - P_high)
70         sigma_high += (P_mean >= P)*(sigma - sigma_high)
71
72     return sigma

```

## variance\_curve.py

```
1 import numpy as np
2
3 Z1 = np.array([0.23934445564954748, 0.2370172145384514,
4               0.23319383855246545, 0.22779527372712297, 0.22410506177795986,
5               0.22796530521676028, 0.22992033119402192, 0.23360387896928214,
6               0.23923251959598327])
7 Z2 = np.array([0.2355916740288041, 0.23547872334450043,
8               0.2278527332290963, 0.2493545607795239, 0.2684664268847795,
9               0.16758709131144714, 0.23526774429356268, 0.16354991037070024,
10              0.09001896821824877])
11 Z3 = np.array([2.3126258447474375, 2.077549483492911,
12              2.1200577204482105, 2.637109433927137, 2.9677374658205307,
13              2.385086643216494, 3.064763505035709, 2.3367542064926443,
14              2.114562825304444])
15
16 # Compute the initial forward variance curve at a given time t
17 # using the Gompertz function with precomputed parameters
18
19 def Gompertz(t, index = 0):
20     z1 = Z1[index]; z2 = Z2[index]; z3 = Z3[index];
21     return z1 * np.exp(-z2 * np.exp(-z3 * t))
22
23 def variance_curve(t, index = 0):
24     z1 = Z1[index]; z2 = Z2[index]; z3 = Z3[index];
25     return (z1 * np.exp(-z2 * np.exp(-z3 * t)))**2 + 2*t*z1**2*z2*
26            z3*np.exp(-2*z2*np.exp(-z3*t)-z3*t)
```

## Heston

### Heston.py

```
1 import numpy as np
2 import QuantLib as ql
3 import scipy.integrate
4 from scipy.special import ive
5
6 ##### PUT-CALL PARITY #####
7
8 def put_call_parity(put, S0, strike, r, q, tau):
9     # Standard put_call_parity
10     return put + S0*np.exp(-q*tau) - strike*np.exp(-r*tau)
11
12 def call_put_parity(call, S0, strike, r, q, tau):
13     return call - S0*np.exp(-q*tau) + strike*np.exp(-r*tau)
14
15
16
17 ##### Analytic Heston #####
18
19 def phi_hest(u, tau, sigma_0, kappa, eta, theta, rho):
20
21     # Compute the characteristic function for Heston Model
22
```

```

23 # u: argument of the function (where you want to evaluate)
24 # tau: time to expiration
25 # sigma_0, kappa, eta, theta, rho: Heston parameters
26
27 alpha_hat = -0.5 * u * (u + 1j)
28 beta = kappa - 1j * u * theta * rho
29 gamma = 0.5 * theta ** 2
30 d = np.sqrt(beta**2 - 4 * alpha_hat * gamma)
31 g = (beta - d) / (beta + d)
32 h = np.exp(-d*tau)
33 A_ = (beta - d)*tau - 2*np.log((g*h-1) / (g-1))
34 A = kappa * eta / (theta**2) * A_
35 B = (beta - d) / (theta**2) * (1 - h) / (1 - g*h)
36 return np.exp(A + B * sigma_0)
37
38 def integral(x, tau, sigma_0, kappa, eta, theta, rho):
39
40     # Pseudo-probabilities
41
42     # x: log-prices discounted
43
44     integrand = (lambda u: np.real(np.exp((1j*u + 0.5)*x) * \
45                                     phi_hest(u - 0.5j, tau, sigma_0,
46                                     kappa, eta, theta, rho)) / \
47                                     (u**2 + 0.25))
48
49     i, err = scipy.integrate.quad_vec(integrand, 0, np.inf)
50
51     return i
52
53 def analytic_hest(S0, strikes, tau, r, q, kappa, theta, rho, eta,
54                 sigma_0, options_type):
55
56     # Pricing of vanilla options under analytic Heston
57
58     a = np.log(S0/strikes) + (r-q)*tau
59     i = integral(a, tau, sigma_0, kappa, eta, theta, rho)
60
61     out = S0 * np.exp(-q*tau) - strikes * np.exp(-r*tau)/np.pi * i
62     out = np.array([out]).flatten()
63
64     for k in range(len(out)):
65         if options_type[k] == 0:
66             out[k] = call_put_parity(out[k], S0, strikes[k], r, q,
67                                     tau)
68
69     return out
70
71 ##### COS METHOD Le Floch
72 #####
73
74 def phi_hest_0(u, tau, r, q, sigma_0, kappa, eta, theta, rho):
75
76     # Compute the characteristic function for Heston Model with
77     log_asset = 0
78
79     # u: argument of the function (where you want to evaluate)

```

```

75     # r: risk-free-rate
76     # q: annual percentage yield
77     # tau: time to expiration
78     # sigma_0, kappa, eta, theta, rho: Heston parameters
79
80     beta = (kappa - 1j*rho*u*theta)
81     d = np.sqrt(beta**2 + (theta**2)*(1j*u+u**2))
82     r_minus = (beta - d)
83     g = r_minus/(beta + d)
84     aux = np.exp(-d*tau)
85
86     term_1 = sigma_0/(theta**2) * ((1-aux)/(1-g*aux)) * r_minus
87     term_2 = kappa*eta/(theta**2) * (tau*r_minus - 2*np.log((1-g*
88     aux) / (1-g)))
89     term_3 = 1j*(r-q)*u*tau
90
91     return np.exp(term_1)*np.exp(term_2)*np.exp(term_3)
92
93 def chi_k(k, c, d, a, b):
94     # Auxiliary function for U_k
95
96     aux_1 = k*np.pi/(b-a)
97     aux_2 = np.exp(d)
98     aux_3 = np.exp(c)
99
100     return (np.cos(aux_1*(d-a))*aux_2 - \
101             aux_3 + \
102             aux_1*np.sin(aux_1*(d-a))*aux_2) / (1+aux_1**2)
103
104 def psi_k(k, c, d, a, b):
105     # Auxiliary function for U_k
106
107     if k == 0:
108         return d - c
109
110     aux = k*np.pi/(b-a)
111     return np.sin(aux*(d-a)) / aux
112
113 def U_k_put(k, a, b):
114     # Auxiliary for cos_method
115
116     return 2./(b-a) * (psi_k(k, a, 0, a, b) - chi_k(k, a, 0, a, b))
117
118 def optimal_ab(r, q, tau, sigma_0, kappa, eta, theta, rho, L = 12):
119     # Compute the optimal interval for the truncation
120     aux = np.exp(-kappa*tau)
121     c1 = (r-q)*tau - sigma_0 * tau / 2
122
123     c2 = (sigma_0) / (4*kappa**3) * (4*kappa**2*(1+(rho*theta*tau
124     -1)*aux) \
125     + kappa*(4*rho*theta*(aux-1)\
126     -2*theta**2*tau*aux) \
127     +theta**2*(1-aux*aux)) \
128     + eta/(8*kappa**3)*(8*kappa**3*tau - 8*kappa**2*(1+ rho*
129     theta*tau +(rho*theta*tau-1)*aux)\
130     + 2*kappa*((1+2*aux)*theta**2*tau+8*(1-
131     aux)*rho*theta)\

```

```

128         + theta**2*(aux*aux+4*aux-5))
129
130
131     return c1 - 12*np.sqrt(np.abs(c2)), c1 + 12*np.sqrt(np.abs(c2))
132
133
134 def precomputed_terms(r, q, tau, sigma_0, kappa, eta, theta, rho, L
135 , N):
136     # Auxiliary term precomputed
137
138     a,b = optimal_ab(r, q, tau, sigma_0, kappa, eta, theta, rho, L)
139     aux = np.pi/(b-a)
140     out = np.zeros(N-1)
141
142     for k in range(1,N):
143         out[k-1] = np.real(np.exp(-1j*k*a*aux)*\
144                             phi_hest_0(k*aux, tau, r, q, sigma_0,
145                             kappa, eta, theta, rho))
146
147     return out, a, b
148
149 def V_k_put(k, a, b, S0, K, z):
150     # V_k coefficients for puts
151
152     return 2./(b-a)*(K*psi_k(k, a, z, a, b) - S0*chi_k(k, a, z, a,
153     b))
154
155 def cos_method_Heston_LF(precomp_term, a, b, tau, r, q, sigma_0,
156 kappa, eta, theta, rho, S0,\
157 strikes, N, options_type, L=12):
158     # Cosine Fourier Expansion for evaluating vanilla options under
159     # Heston using LeFloch correction
160     # Should be better for deep otm options.
161
162     # precomp_term: precomputed terms from the function
163     precomputed_terms
164     # a,b: extremes of the interval to approximate
165     # tau: time to expiration (annualized) (must be a number)
166     # r: risk-free-rate
167     # q: yield
168     # sigma_0, kappa, eta, theta, rho: Heston parameters
169     # S0: initial spot price
170     # strikes: np.array of strikes
171     # N: number of terms of the truncated expansion
172     # options_type: binary np.array (1 for calls, 0 for puts)
173     # L: truncation level
174
175     z = np.log(strikes/S0)
176
177     out = 0.5 * np.real(phi_hest_0(0, tau, r, q, sigma_0, kappa,
178     eta, theta, rho))*\
179         V_k_put(0, a, b, S0, strikes, z)
180
181     for k in range(1,N):
182         out = out + precomp_term[k-1]*V_k_put(k, a, b, S0, strikes,
183         z)

```

```

177 D = np.exp(-r*tau)
178 out = out*D
179
180 for k in range(len(strikes)):
181     if options_type[k] == 1:
182         out[k] = put_call_parity(out[k], S0, strikes[k], r, q,
183 tau)
184
185 return out
186
187 #####Calibration
188 #####
189
190 def setup_model(_yield_ts, _dividend_ts, _spot,
191 init_condition):
192     # Setup Heston model object
193
194     # _yield_ts: Term Structure for yield (QuantLib object)
195     # _dividend_ts: Term Structure for dividend_ts (QuantLib object)
196     # init_condition: eta, kappa, theta, rho, sigma_0
197
198     eta, kappa, theta, rho, sigma_0 = init_condition
199     process = ql.HestonProcess(_yield_ts, _dividend_ts,
200 ql.QuoteHandle(ql.SimpleQuote(_spot)),
201 sigma_0, kappa, eta, theta, rho)
202     model = ql.HestonModel(process)
203     engine = ql.AnalyticHestonEngine(model)
204     return model, engine
205
206 def setup_helpers(engine, expiration_dates, strikes,
207 data, ref_date, spot, yield_ts,
208 dividend_ts, calendar):
209     # Helpers for Heston Calibration
210
211     # engine: Heston.setup_model output
212     # expiration_dates: maturities
213     # data: IV market data
214     # ref_date: date for the calculation
215     # yield_ts: Term Structure for yield (QuantLib object)
216     # dividend_ts: Term Structure for dividend_ts (QuantLib object)
217     # calendar: type of calendar for calculations
218
219     heston_helpers = []
220     grid_data = []
221     for i, date in enumerate(expiration_dates):
222         for j, s in enumerate(strikes):
223             t = (date - ref_date)
224             p = ql.Period(t, ql.Days)
225             vols = data[i][j]
226             helper = ql.HestonModelHelper(
227 p, calendar, spot, s,
228 ql.QuoteHandle(ql.SimpleQuote(vols)),
229 yield_ts, dividend_ts)
230             helper.setPricingEngine(engine)
231             heston_helpers.append(helper)
232             grid_data.append((date, s))

```



```

231     return heston_helpers, grid_data
232
233 def cost_function_generator(model, helpers, norm=False):
234     # Define cost function for the calibration (usually Mean Square
235     # Error)
236
237     def cost_function(params):
238         params_ = ql.Array(list(params))
239         model.setParams(params_)
240         error = [h.calibrationError() for h in helpers]
241         if norm:
242             return np.sqrt(np.sum(np.abs(error)))
243         else:
244             return error
245     return cost_function
246
247 #####Simulation Heston#####
248
249 def create_totems(base, start, end):
250     # create the grid
251
252     totems = np.ones(end-start+2)
253     index = 1
254     for j in range(start, end+1):
255         totems[index] = base**j
256         index += 1
257
258     totems[0] = 0
259     return totems
260
261 def calc_nu_bar(kappa, eta, theta):
262     # compute v bar
263     return 4*kappa*eta/theta**2
264
265 def x2_exp_var(nu_bar, kappa, theta, dt):
266     # compute E[X_2] and Var[X_2]
267
268     aux = kappa*dt/2.
269     c1 = np.cosh(aux)/np.sinh(aux)
270     c2 = (1./np.sinh(aux))**2
271     exp_x2 = nu_bar*theta**2*((-2.+kappa*dt*c1)/(4*kappa**2))
272     var_x2 = nu_bar*theta**4*((-8.+2*kappa*dt*c1+\
273         kappa**2*dt**2*c2)/(8*kappa**4))
274     return exp_x2, var_x2
275
276 def Z_exp_var(nu_bar, exp_x2, var_x2):
277     # compute E[Z] and Var[Z]
278
279     return 4*exp_x2/nu_bar, 4*var_x2/nu_bar
280
281 def xi_exp(nu_bar, kappa, theta, dt, totem):
282     # compute E[\Xi] and E[\Xi^2]
283
284     z = 2*kappa*np.sqrt(totem) / (theta**2*np.sinh(kappa*dt/2.))
285     iv_pre = ive(nu_bar/2.-1., z)
286     exp_xi = (z*ive(nu_bar/2., z))/(2*iv_pre)

```

```

286     exp_xi2 = exp_xi + (z**2*ive(nu_bar/2.+1,z))/(4.*iv_pre)
287
288     return exp_xi, exp_xi2
289
290 def create_caches(base, start, end, kappa, eta, theta, dt):
291     # precompute the caches for IV*
292
293     totems = create_totems(base, start, end)
294     caches_exp = np.zeros(end-start+2)
295     caches_var = np.zeros(end-start+2)
296     nu_bar = calc_nu_bar(kappa, eta, theta)
297     exp_x2, var_x2 = x2_exp_var(nu_bar, kappa, theta, dt)
298     exp_Z, var_Z = Z_exp_var(nu_bar, exp_x2, var_x2)
299
300     for j in range(1, end-start+2):
301         exp_xi, exp_xi2 = xi_exp(nu_bar, kappa, theta, dt, totems[j]
302         )
303         caches_exp[j] = exp_x2 + exp_xi*exp_Z
304         caches_var[j] = var_x2 + exp_xi*var_Z + \
305             (exp_xi2-exp_xi**2)*exp_Z**2
306
307     caches_exp[0] = exp_x2
308     caches_var[0] = var_x2
309     return totems, caches_exp, caches_var
310
311 def x1_exp_var(kappa, theta, dt, vt, vT):
312     # compute E[X_1] and Var[X_1]
313
314     aux = kappa*dt/2.
315     c1 = np.cosh(aux)/np.sinh(aux)
316     c2 = (1./np.sinh(aux))**2
317
318     exp_x1 = (vt + vT)*(c1/kappa - dt*c2/2)
319     var_x1 = (vt + vT)*theta**2*(c1/kappa**3 + dt*c2/(2*kappa**2) \
320         - dt**2*c1*c2/(2*kappa))
321
322     return exp_x1, var_x1
323
324 def lin_interp(vtvT, totems, caches_exp, caches_var):
325     # compute linear interpolation for value not in caches
326
327     exp_int = np.interp(vtvT, totems, caches_exp)
328     var_int = np.interp(vtvT, totems, caches_var)
329     return exp_int, var_int
330
331 def sample_vT(vt, dt, kappa, theta, nu_bar):
332     # sample vT from a noncentral chisquare (given vt)
333
334     aux = (theta**2*(1-np.exp(-kappa*dt)))/(4*kappa)
335     n = np.exp(-kappa*dt)/aux *vt
336     return np.random.noncentral_chisquare(nu_bar, n)*aux
337
338 def generate_path(S0, T, dt, kappa, eta, theta, rho, r, q, sigma_0,
339     totems, caches_exp, caches_var):
340     # This function generate one path for the Heston model using
341     # the Gamma Approx algorithm

```

```

340 # T: final time
341 # r: risk-free-rate
342 # q: yield
343 # sigma_0, kappa, eta, theta, rho: Heston parameters
344 # S0: initial spot price
345 # dt: temporal step
346 # totems, caches_exp, caches_var: precomputed grid, caches for
    expectation and caches for var
347
348 t = dt
349 index = 0
350 vt = sigma_0
351 xt = np.log(S0)
352
353 path = np.zeros(int(np.ceil(T/dt))+1)
354 path[0] = S0
355
356 variance = np.zeros(int(np.ceil(T/dt))+1)
357 variance[0] = vt
358
359 nu_bar = calc_nu_bar(kappa, eta, theta)
360
361 while t < T:
362     vT = sample_vT(vt, dt, kappa, theta, nu_bar)
363
364     exp_int, var_int = lin_interp(vt*vT, totems, caches_exp,
        caches_var)
365
366     exp_x1, var_x1 = x1_exp_var(kappa, theta, dt, vt, vT)
367     exp_int += exp_x1
368     var_int += var_x1
369
370     gamma_t = var_int/exp_int
371     gamma_k = exp_int**2/var_int
372
373     iv_t = np.random.gamma(gamma_k, gamma_t)
374     z = np.random.normal()
375
376     xt += (r-q)*dt + (- 0.5 + kappa*rho/theta)*iv_t + \
        rho/theta*(vT-vt-kappa*eta*dt) + \
377     z*np.sqrt(1-rho**2)*np.sqrt(iv_t)
378
379     index += 1
380     path[index] = np.exp(xt)
381     vt = vT
382     variance[index] = vt
383     t += dt
384
385
386 return path[:-1], variance[:-1]

```

## Rough Heston

### rHeston.py

```

1 import numpy as np
2 import ImpliedDrift

```

```

3 import Heston
4 import BlackScholes
5 import scipy.integrate
6
7 from variance_curve import variance_curve, Gompertz
8 from scipy.special import gamma
9 from scipy.interpolate import CubicSpline
10 from scipy.stats import norm
11
12 du = 1e-4
13 GRID = np.linspace(0,10,int(1./du))
14
15
16 ##### Pade rHeston
17 #####
18 def Pade33(u, t, H, rho, theta):
19     alpha = H + 0.5
20
21     aa = np.sqrt(u * (u + (0+1j)) - rho**2 * u**2)
22     rm = -(0+1j) * rho * u - aa
23     rp = -(0+1j) * rho * u + aa
24
25     gamma1 = gamma(1+alpha)
26     gamma2 = gamma(1+2*alpha)
27     gammam1 = gamma(1-alpha)
28     gammam2 = gamma(1-2*alpha)
29
30     b1 = -u*(u+1j)/(2 * gamma1)
31     b2 = (1-u*1j) * u**2 * rho/(2* gamma2)
32     b3 = gamma2/gamma(1+3*alpha) * \
33         (u**2*(1j+u)**2/(8*gamma1**2)+(u+1j)*u**3*rho**2/(2*gamma2)
34         )
35
36     g0 = rm
37     g1 = -rm/(aa*gammam1)
38     g2 = rm/aa**2/gammam2 * \
39         (1 + rm/(2*aa)*gammam2/gammam1**2)
40
41     den = g0**3 +2*b1*g0*g1-b2*g1**2+b1**2*g2+b2*g0*g2
42
43     p1 = b1
44     p2 = (b1**2*g0**2 + b2*g0**3 + b1**3*g1 + b1*b2*g0*g1 - \
45         b2**2*g1**2 +b1*b3*g1**2 +b2**2*g0*g2 - b1*b3*g0*g2)/den
46     q1 = (b1*g0**2 + b1**2*g1 - b2*g0*g1 + b3*g1**2 - b1*b2*g2 -b3*
47         g0*g2)/den
48     q2 = (b1**2*g0 + b2*g0**2 - b1*b2*g1 - b3*g0*g1 + b2**2*g2 - b1
49         *b3*g2)/den
50     q3 = (b1**3 + 2*b1*b2*g0 + b3*g0**2 -b2**2*g1 +b1*b3*g1 )/den
51     p3 = g0*q3
52
53     y = t**alpha
54
55     return (p1*y + p2*y**2 + p3*y**3)/(1 + q1*y + q2*y**2 + q3*y
56         **3)
57
58 def DH_Pade33(u, x, H, rho, theta):

```

```

55     alpha = H + 0.5
56
57     aa = np.sqrt(u * (u + 1j) - rho**2 * u**2)
58     rm = -1j * rho * u - aa
59     rp = -1j * rho * u + aa
60
61     b1 = -u*(u+1j)/(2 * gamma(1+alpha))
62     b2 = (1-u*1j) * u**2 * rho/(2* gamma(1+2*alpha))
63     b3 = gamma(1+2*alpha)/gamma(1+3*alpha) * \
64         (u**2*(1j+u)**2/(8*gamma(1+alpha)**2)+(u+1j)*u**3*rho
        **2/(2*gamma(1+2*alpha)))
65
66     g0 = rm
67     g1 = -rm/(aa*gamma(1-alpha))
68     g2 = rm/aa**2/gamma(1-2*alpha) * (1 + rm/(2*aa)*gamma(1-2*alpha)
        )/gamma(1-alpha)**2)
69
70     den = g0**3 +2*b1*g0*g1-b2*g1**2+b1**2*g2+b2*g0*g2
71
72     p1 = b1
73     p2 = (b1**2*g0**2 + b2*g0**3 + b1**3*g1 + b1*b2*g0*g1 - b2**2*
        g1**2 + b1*b3*g1**2 +b2**2*g0*g2 - b1*b3*g0*g2)/den
74     q1 = (b1*g0**2 + b1**2*g1 - b2*g0*g1 + b3*g1**2 - b1*b2*g2 -b3*
        g0*g2)/den
75     q2 = (b1**2*g0 + b2*g0**2 - b1*b2*g1 - b3*g0*g1 + b2**2*g2 - b1
        *b3*g2)/den
76     q3 = (b1**3 + 2*b1*b2*g0 + b3*g0**2 -b2**2*g1 +b1*b3*g1 )/den
77     p3 = g0*q3
78
79     y = x**alpha
80
81     hpade = (p1*y + p2*y**2 + p3*y**3)/(1 + q1*y + q2*y**2 + q3*y
        **3)
82
83     res = 0.5*(hpade-rm)*(hpade-rp)
84
85     return res
86
87 def phi_rhest(u, t, H, rho, theta):
88     if u == 0:
89         return 1.
90
91     N = int(t*365)
92     alpha = H + 0.5
93     dt = t/N
94     tj = np.linspace(0,N,N+1,endpoint = True)*dt
95
96     x = theta**(1./alpha)*tj
97     xi = np.flip(variance_curve(tj))
98
99     aux = DH_Pade33(u, x, H, rho, theta)
100
101     return np.exp(np.matmul(aux,xi)*dt)
102
103 def DH_Pade33_vec(u, x, H, rho, theta):
104     alpha = H + 0.5
105

```

```

106 aa = np.sqrt(u * (u + 1j) - rho**2 * u**2)
107 rm = -1j * rho * u - aa
108 rp = -1j * rho * u + aa
109
110 b1 = -u*(u+1j)/(2 * gamma(1+alpha))
111 b2 = (1-u*1j) * u**2 * rho/(2* gamma(1+2*alpha))
112 b3 = gamma(1+2*alpha)/gamma(1+3*alpha) * \
113      (u**2*(1j+u)**2/(8*gamma(1+alpha)**2)+(u+1j)*u**3*rho
      **2/(2*gamma(1+2*alpha)))
114
115 g0 = rm
116 g1 = -rm/(aa*gamma(1-alpha))
117 g2 = rm/aa**2/gamma(1-2*alpha) * (1 + rm/(2*aa)*gamma(1-2*alpha)
    )/gamma(1-alpha)**2)
118
119 den = g0**3 +2*b1*g0*g1-b2*g1**2+b1**2*g2+b2*g0*g2
120
121 p1 = b1
122 p2 = (b1**2*g0**2 + b2*g0**3 + b1**3*g1 + b1*b2*g0*g1 - b2**2*
    g1**2 +b1*b3*g1**2 +b2**2*g0*g2 - b1*b3*g0*g2)/den
123 q1 = (b1*g0**2 + b1**2*g1 - b2*g0*g1 + b3*g1**2 - b1*b2*g2 -b3*
    g0*g2)/den
124 q2 = (b1**2*g0 + b2*g0**2 - b1*b2*g1 - b3*g0*g1 + b2**2*g2 - b1
    *b3*g2)/den
125 q3 = (b1**3 + 2*b1*b2*g0 + b3*g0**2 -b2**2*g1 +b1*b3*g1 )/den
126 p3 = g0*q3
127
128 y = x**alpha
129 y2 = y**2
130 y3 = y**3
131
132 size_ = len(u)
133 Y = np.tile(y, (size_,1)).transpose()
134 Y2 = np.tile(y2, (size_,1)).transpose()
135 Y3 = np.tile(y3, (size_,1)).transpose()
136
137 hpade = (Y*p1 + Y2*p2 + Y3*p3)/(1 + Y*q1 + Y2*q2 + Y3*q3)
138
139 res = 0.5*(hpade-rm)*(hpade-rp)
140
141 return res
142
143 def phi_rhest_vec(u, t, H, rho, theta, N = 1000):
144
145     mask = (u == 0)
146
147     alpha = H + 0.5
148     dt = t/N
149     tj = np.linspace(0,N,N+1,endpoint = True)*dt
150
151     x = theta**(1./alpha)*tj
152     xi = np.flip(variance_curve(tj))
153
154     res = np.zeros(len(u), dtype = complex)
155
156     if mask.any():
157         aux = DH_Pade33_vec(u[~mask], x, H, rho, theta)

```

```

158     res[~mask] = np.exp(np.matmul(xi,aux)*dt)
159     res[mask] = 1.
160     else:
161         aux = DH_Pade33_vec(u, x, H, rho, theta)
162         res = np.exp(np.matmul(xi,aux)*dt)
163
164     return res
165
166 # ##### Analytic rHeston
167 # #####
168 def integral(x, t, H, rho, theta):
169
170     integrand = (lambda u: np.real(np.exp((1j*u)*x) * \
171                                     phi_rhest(u - 0.5j, t, H, rho,
172                                     theta)) / \
173                 (u**2 + 0.25))
174
175     i, err = scipy.integrate.quad_vec(integrand, 0, np.inf)
176
177     return i
178
179 # def integral_vec(x, t, H, rho, theta, grid = GRID):
180 #     aux = (np.tile(grid, (len(x),1)).transpose()*x).transpose()
181 #     i = np.real(np.exp(1j*aux)*phi_rhest_vec(grid - 0.5j, t, H,
182 #     rho, theta)) / \
183 #         (grid**2 + 0.25)
184 #     i = i.sum(axis = 1)*(grid[1]-grid[0])
185 #     return i
186
187 def analytic_rhest(S0, strikes, t, H, rho, theta, options_type):
188
189     # Pricing of vanilla options under "analytic" rHeston using
190     # Lewis Formula
191
192     a = np.log(S0/strikes) + ImpliedDrift.drift(t)*t
193     i = integral(a, t, H, rho, theta)
194     r = ImpliedDrift.r(t)
195     q = ImpliedDrift.q(t)
196     out = S0 * np.exp(-q*t) - np.sqrt(S0*strikes) * np.exp(-(r+q)*t
197     *0.5)/np.pi * i
198     out = np.array([out]).flatten()
199
200     for k in range(len(options_type)):
201         if options_type[k] == 0:
202             out[k] = Heston.call_put_parity(out[k], S0, strikes[k],
203             r, q, t)
204
205     if (out < 0).any():
206         out[out < 0] = 0.
207
208     return out
209
210 # def analytic_rhest_vec(S0, strikes, t, H, rho, theta,
211 # options_type):

```

```

208
209 # # Pricing of vanilla options under "analytic" rHeston using
    Lewis Formula
210
211 # a = np.log(S0/strikes) + ImpliedDrift.drift(t)*t
212 # i = integral_vec(a, t, H, rho, theta)
213 # r = ImpliedDrift.r(t)
214 # q = ImpliedDrift.q(t)
215 # out = S0 * np.exp(-q*t) - np.sqrt(S0*strikes) * np.exp(-(r+q)
    *t*0.5)/np.pi * i
216 # out = np.array([out]).flatten()
217
218 # for k in range(len(options_type)):
219 #     if options_type[k] == 0:
220 #         out[k] = Heston.call_put_parity(out[k], S0, strikes[k]
    ], r, q, t)
221
222 # if (out < 0).any():
223 #     out[out < 0] = 0.
224
225 # return out
226
227 #####Simulation rHeston
    #####
228
229 # Psi for the QE Scheme of Lemma 7.
230 def psi_m(psi, ev, w):
231     #psi minus
232
233     beta2 = psi
234     mask = psi > 0
235     mask1 = psi <= 0
236     if np.any(mask):
237         beta2[mask] = 2./psi[mask]-1+np.sqrt(2./psi[mask]* \
    np.abs(2./psi[mask]-1)
238     )
239     if np.any(mask1):
240         beta2[mask1] = 0.
241     return ev/(1+beta2)*(np.sqrt(np.abs(beta2))+w)**2
242
243 def psi_p(psi, ev, u):
244     #psi plus
245
246     p = 2/(1+psi)
247     res = (u<p)*(-ev)/2*(1+psi)
248     mask = u > 0
249     if np.any(mask):
250         res[mask] = np.log(u[mask]/p[mask])
251     return res
252
253 # functions for K_i, K_ii and K_01
254 def Gi(eta, alpha, dt, i):
255     return np.sqrt(2*alpha-1)*eta/alpha * dt**alpha * ((i+1)**alpha
    - i**alpha)
256
257 def Gii(eta, H, dt, i):
258     aux = 2*H

```



```

259     return eta**2 * dt**aux * ((i+1)**aux - i**aux)
260
261 def G01(eta, alpha, dt):
262     return Gi(eta,alpha,dt,0)*Gi(eta,alpha,dt,1)/dt
263
264 def HQE_sim(theta, H, rho, T, S0, paths, steps, eps0 = 1e-10):
265     # HQE scheme
266
267     # theta, H, rho: parameters of the rHeston model
268     # T: final time of the simulations, in years
269     # S0: spot price at time 0
270     # paths: number of paths to simulate
271     # steps: number of timesteps between 0 and T
272     # eps0: lower bound for xihat
273
274     dt = T/steps
275     dt_sqrt = np.sqrt(dt)
276     alpha = H + 0.5
277     eta = theta/(gamma(alpha)*np.sqrt(2*H))
278     rho2m1 = np.sqrt(1-rho*rho)
279
280     W = np.random.normal(0.,1.,size = (steps,paths))
281     Wperp = np.random.normal(0.,1.,size = (steps,paths))
282     Z = np.random.normal(0.,1.,size = (steps,paths))
283     U = np.random.uniform(0.,1.,size = (steps,paths))
284     Uperp = np.random.uniform(0.,1.,size = (steps,paths))
285
286     tj = np.arange(0,steps,1)*dt
287     tj += dt
288
289     xij = variance_curve(tj)
290     G0del = Gi(eta,alpha,dt,0)
291     G00del = Gii(eta,alpha,dt,0)
292     G11del = Gii(eta,alpha,dt,1)
293     G01del = G01(eta,alpha,dt)
294     G00j = np.zeros(steps)
295
296     for j in range(steps):
297         G00j[j] = Gii(eta,H,dt,j)
298     bstar = np.sqrt((G00j)/dt)
299
300     rho_vchi = G0del/np.sqrt(G00del*dt)
301     beta_vchi = G0del/dt
302
303     u = np.zeros((steps,paths))
304     chi = np.zeros((steps,paths))
305     v = np.ones(paths)*variance_curve(0)
306     hist_v = np.zeros((steps,paths))
307     hist_v[0,:] = v
308     xihat = np.ones(paths)*xij[0]
309     x = np.zeros((steps,paths))
310     y = np.zeros(paths)
311     w = np.zeros(paths)
312
313     for j in range(steps):
314         xibar = (xihat + 2*H*v)/(1+2*H)
315

```

```

316     psi_chi = 2*beta_vchi*xibar*dt/(xihat**2)
317     psi_eps = 2/(xihat**2)*xibar*(G00del - G0del**2/dt)
318     aux_ = xihat/2
319
320     z_chi = np.zeros(paths)
321     z_eps = np.zeros(paths)
322
323     mask1 = psi_chi < 1.5
324     mask2 = psi_chi >= 1.5
325     mask3 = psi_eps < 1.5
326     mask4 = psi_eps >= 1.5
327
328     if np.any(mask1):
329         z_chi[mask1] = psi_m(psi_chi[mask1], aux_[mask1], W[j,
mask1])
330     if np.any(mask2):
331         z_chi[mask2] = psi_m(psi_chi[mask2], aux_[mask2], U[j,
mask2])
332     if np.any(mask3):
333         z_eps[mask3] = psi_m(psi_eps[mask3], aux_[mask3], Wperp[j
,mask3])
334     if np.any(mask4):
335         z_eps[mask4] = psi_m(psi_eps[mask4], aux_[mask4], Uperp[j
,mask4])
336
337     chi[j,:] = (z_chi-aux_)/beta_vchi
338     eps = z_eps - aux_
339     u[j,:] = beta_vchi*chi[j,:]+eps
340     vf = xihat + u[j,:]
341     vf[vf < eps0] = eps0
342
343     dw = (v+vf)/2*dt
344     w += dw
345     y += chi[j,:]
346     x[j,:] = x[j-1,:] + ImpliedDrift.drift(T)*dt - dw/2 + np.
sqrt(dw) \
347         * (rho2m1*Z[j,:]) + rho*chi[j,:]
348
349     btilde = np.flip(bstar[1:j+1])
350     if j < steps-1:
351         xihat = xij[j+1] + (np.matmul(btilde, chi[:j,:]))
352     v = vf
353     hist_v[j,:] = v
354     return np.vstack([np.ones(paths)*S0, (np.exp(x)*S0)]), hist_v

```

## Rough Bergomi

### utils\_rBergomi.py

```

1 import numpy as np
2
3 # TBSS kernel applicable to the rBergomi variance process.
4 def g(x, a):
5     return x**a
6

```

```

7 # Optimal discretisation of TBSS process for minimising hybrid
  scheme error.
8 def b(k, a):
9     return ((k**(a+1)-(k-1)**(a+1))/(a+1))*(1/a)
10
11 # Covariance matrix for given alpha and n, assuming kappa = 1.
12 def cov(a, n):
13     cov = np.array([[0.,0.],[0.,0.]])
14     cov[0,0] = 1./n
15     cov[0,1] = 1./((1.*a+1) * n**(1.*a+1))
16     cov[1,1] = 1./((2.*a+1) * n**(2.*a+1))
17     cov[1,0] = cov[0,1]
18     return cov

```

## rbergomi.py

```

1 import numpy as np
2 from scipy.signal import convolve
3 from numpy.random import default_rng
4 from utils_rBergomi import *
5 import ImpliedDrift as iD
6
7 # Class for generating paths of the rBergomi model.
8 class rBergomi(object):
9
10     def __init__(self, n, N, T, a):
11
12         # Basic assignments
13         self.T = T
14         # Maturity
15         self.n = n
16         # Steps per year
17         self.dt = 1.0/self.n
18         # Step size
19         self.s = np.round(self.n * self.T).astype(int)
20         # Number of total steps
21         self.t = np.linspace(0, self.T, 1 + self.s)[np.newaxis,:]
22         # Time grid
23         self.a = a
24         # Alpha
25         self.N = N
26         # Number of paths
27
28         # Construct hybrid scheme correlation structure with kappa
29         = 1
30         self.e = np.array([0,0])
31         self.c = cov(self.a, self.n)
32
33     def dW1(self):
34         np.random.seed(0)
35         # Produces random numbers for variance process with
36         required covariance structure
37         N = int(self.N/2)
38         w = np.random.multivariate_normal(self.e, self.c, (N, self.
39 s))
40         return np.concatenate((w,-w), axis = 0)
41

```

```

32 def dW2(self):
33     np.random.seed(0)
34     #Obtain orthogonal increments
35     N = int(self.N/2)
36     w = np.random.randn(N, self.s) * np.sqrt(self.dt)
37     return np.concatenate((w,-w), axis = 0)
38
39 def Y(self, dW):
40     #Constructs Volterra process from appropriately correlated
41     2d Brownian increments
42
43     Y1 = np.zeros((self.N, 1 + self.s)) # Exact integrals
44     Y2 = np.zeros((self.N, 1 + self.s)) # Riemann sums
45
46     Y1[:,1 : self.s+1] = dW[:, :self.s, 1] # Assumes kappa =
47     1
48
49     # Construct arrays for convolution
50     G = np.zeros(1 + self.s) # Gamma
51     for k in np.arange(2, 1 + self.s, 1):
52         G[k] = g(b(k, self.a)/self.n, self.a)
53
54     X = dW[:, :, 0] # Xi
55
56     # Compute convolution and extract relevant terms
57     for i in range(self.N):
58         Y2[i, :] = np.convolve(G, X[i, :])[:1+self.s]
59
60     # Finally construct and return full process
61     return np.sqrt(2 * self.a + 1) * (Y1 + Y2)
62
63 def dZ(self, dW1, dW2, rho):
64     # Constructs correlated price Brownian increments, dB
65
66     self.rho = rho
67     return rho * dW1[:, :, 0] + np.sqrt(1 - rho**2) * dW2
68
69 def V(self, Y, xi, eta):
70     # rBergomi variance process.
71     self.xi = xi
72     self.eta = eta
73     a = self.a
74     t = self.t
75     return xi * np.exp(eta * Y - 0.5 * eta**2 * t**(2 * a + 1))
76     #return xi * ne.evaluate('exp(eta * Y - 0.5 * eta**2 * t
77     *(2 * a + 1))')
78
79 def S_all_path(self, V, dZ, r, q, S0):
80     # rBergomi price process.
81     self.S0 = S0
82     dt = self.dt
83     rho = self.rho
84
85     # Construct non-anticipative Riemann increments
86     increments = np.sqrt(V[:, :-1]) * dZ - 0.5 * V[:, :-1] * dt +
87     (r - q) * dt
88     integral = np.cumsum(increments, axis = 1)

```

```

85
86     S = np.zeros_like(V)
87     S[:,0] = S0
88     S[:,1:] = S0 * np.exp(integral)
89     return S
90
91     def S(self, V, dZ, r, q, S0):
92         # rBergomi price process.
93         self.S0 = S0
94         dt = self.dt
95         rho = self.rho
96
97         # Construct non-anticipative Riemann increments
98         exponent = np.zeros(self.N)
99         for i in range(self.s):
100             exponent += np.sqrt(V[:,i]) * dZ[:,i] - 0.5 * V[:,i] *
dt + (r - q) * dt
101
102         return S0 * np.exp(exponent)
103
104     def global_S(self, V, dZ, S0, steps, index = 0):
105         # rBergomi price process.
106         self.S0 = S0
107         dt = self.dt
108         rho = self.rho
109
110         r = iD.r(self.t[0], index)
111         q = iD.q(self.t[0], index)
112
113         S = list()
114         logS = np.log(S0)
115         for i in range(self.s):
116             logS += np.sqrt(V[:,i]) * dZ[:,i] - 0.5 * V[:,i] * dt +
(r[i] - q[i]) * dt
117             if i in steps:
118                 S.append(np.exp(logS))
119
120         S.append(np.exp(logS))
121
122         return np.array(S)

```

## Quintic Ornstein-Uhlenbeck

```

1 import numpy as np
2 import variance_curve as vc
3 import ImpliedDrift as iD
4 import scipy
5 import BlackScholes as bs
6
7 from scipy.integrate import quad
8
9 def horner_vector(poly, n, x):
10     #Initialize result
11     result = poly[0].reshape(-1,1)
12     for i in range(1,n):
13         result = result*x + poly[i].reshape(-1,1)

```

```

14     return result
15
16
17
18 def gauss_dens(mu,sigma,x):
19     return 1/np.sqrt(2*np.pi*sigma**2)*np.exp(-(x-mu)**2/(2*sigma
20     **2))
21
22
23 def vix_futures(H, eps, T, a_k_part, k, r, q, n_steps, index = 0):
24
25     a2,a4 = (0,0)
26     a0,a1,a3,a5 = a_k_part
27     a_k = np.array([a0, a1, a2, a3, a4, a5])
28
29     kappa_tild = (0.5-H)/eps
30     eta_tild = eps**(H-0.5)
31
32     deltt = 30/365
33     T_delta = T + deltt
34
35     std_X = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
36     kappa_tild*T)))
37     dt = deltt/(n_steps)
38     tt = np.linspace(T, T_delta, n_steps+1)
39
40     FV_curve_all_vix = vc.variance_curve(tt, index)
41
42     exp_det = np.exp(-kappa_tild*(tt-T))
43     cauchy_product = np.convolve(a_k,a_k)
44
45     std_Gs_T = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
46     kappa_tild*(tt-T))))
47     std_X_t = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
48     kappa_tild*tt)))
49     std_X_T = std_X
50
51     n = len(a_k)
52
53     normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].reshape
54     (-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
55     scipy.special.factorial2(np.arange(0,2*n,2).reshape(-1,1)-1),
56     axis=0) #g(u)
57
58     beta = []
59     for i in range(0,2*n-1):
60         k_array = np.arange(i,2*n-1)
61         beta_temp = ((std_Gs_T**((k_array-i).reshape(-1,1))*((
62         k_array-i-1)%2).reshape(-1,1))*\
63         scipy.special.factorial2(k_array-i-1).reshape(-1,1))*\
64         (scipy.special.comb(k_array,i).reshape(-1,1))*\
65         exp_det**(i))*cauchy_product[k_array].reshape(-1,1)
66         beta.append(np.sum(beta_temp,axis=0))
67
68     beta = np.array(beta)*FV_curve_all_vix/normal_var
69     beta = (np.sum((beta[:, :-1]+beta[:, 1:])/2,axis=1))*dt

```

```

64
65     sigma = np.sqrt(eps**(2*H)/(1-2*H)*(1-np.exp((2*H-1)*T/eps)))
66
67     f = lambda x: np.sqrt(horner_vector(beta[:-1], len(beta), x)/
68                             delt)*100 * gauss_dens(0, sigma, x)
69
70     Ft, err = quad(f, -np.inf, np.inf)
71
72     return Ft * np.exp((r-q)*T)
73
74
75 def vix_iv(H, eps, T, a_k_part, K, r, q, n_steps, index = 0):
76
77     a2,a4 = (0,0)
78     a0,a1,a3,a5 = a_k_part
79     a_k = np.array([a0, a1, a2, a3, a4, a5])
80
81     kappa_tild = (0.5-H)/eps
82     eta_tild = eps**(H-0.5)
83
84     delt = 30/365
85     T_delta = T + delt
86
87     std_X = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
88                             kappa_tild*T)))
89     dt = delt/(n_steps)
90     tt = np.linspace(T, T_delta, n_steps+1)
91
92     FV_curve_all_vix = vc.variance_curve(tt, index)
93
94     exp_det = np.exp(-kappa_tild*(tt-T))
95     cauchy_product = np.convolve(a_k,a_k)
96
97     std_Gs_T = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
98                             kappa_tild*(tt-T))))
99     std_X_t = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
100                             kappa_tild*tt)))
101     std_X_T = std_X
102
103     n = len(a_k)
104
105     normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].reshape
106                         (-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
107                         scipy.special.factorial2(np.arange(0,2*n,2).reshape(-1,1)-1),
108                         axis=0) #g(u)
109
110     beta = []
111     for i in range(0,2*n-1):
112         k_array = np.arange(i,2*n-1)
113         beta_temp = ((std_Gs_T**((k_array-i).reshape(-1,1))*((

```

```

114     beta = np.array(beta)*FV_curve_all_vix/normal_var
115     beta = (np.sum((beta[:, :-1]+beta[:, 1:])/2,axis=1))*dt
116
117     sigma = np.sqrt(eps**(2*H)/(1-2*H)*(1-np.exp((2*H-1)*T/eps)))
118
119     N = len(K); P = np.zeros(N);
120
121     for i in range(N):
122
123         f = lambda x: np.maximum(np.sqrt(horner_vector(beta[:, :-1],
124             len(beta), x)/delt)*100 - K[i], 0) * gauss_dens(0, sigma, x)
125         P[i], err = quad(f, -np.inf, np.inf)
126
127     return P * np.exp((r-q)*T)
128
129
130 def dW(n_steps, N_sims):
131     w = np.random.normal(0, 1, (n_steps, N_sims))
132     #Antithetic variates
133     w = np.concatenate((w, -w), axis = 1)
134     return w
135
136
137
138 def local_reduction(rho,H,eps,T,a_k_part,S0,strike_array,n_steps,
139     N_sims,w1,r,q, index = 0):
140
141     eta_tild = eps**(H-0.5)
142     kappa_tild = (0.5-H)/eps
143
144     a_0,a_1,a_3,a_5 = a_k_part
145     a_k = np.array([a_0,a_1,0,a_3,0,a_5])
146
147     dt = T/n_steps
148     tt = np.linspace(0., T, n_steps + 1)
149
150     exp1 = np.exp(kappa_tild*tt)
151     exp2 = np.exp(2*kappa_tild*tt)
152
153     diff_exp2 = np.concatenate((np.array([0.]),np.diff(exp2)))
154     std_vec = np.sqrt(diff_exp2/(2*kappa_tild))[:,np.newaxis] #to
155     #be broadcasted columnwise
156     exp1 = exp1[:,np.newaxis]
157     X = (1/exp1)*(eta_tild*np.cumsum(std_vec*w1, axis = 0))
158     Xt = np.array(X[:-1])
159     del X
160
161     tt = tt[:-1]
162     std_X_t = np.sqrt(eta_tild**2/(2*kappa_tild)*(1-np.exp(-2*
163         kappa_tild*tt)))
164     n = len(a_k)
165
166     cauchy_product = np.convolve(a_k,a_k)
167     normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].reshape
168         (-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
169         scipy.special.factorial2(np.arange(0,2*n,2).reshape(-1,1)

```



```

-1),axis=0)
166
167 f_func = horner_vector(a_k[::-1], len(a_k), Xt)
168
169 del Xt
170
171 fv_curve = vc.variance_curve(tt, index).reshape(-1,1)
172
173 volatility = f_func/np.sqrt(normal_var.reshape(-1,1))
174 del f_func
175 volatility = np.sqrt(fv_curve)*volatility
176
177 logS1 = np.log(S0)
178 for i in range(w1.shape[0]-1):
179     logS1 = logS1 - 0.5*dt*(volatility[i]*rho)**2 + np.sqrt(dt)
    *rho*volatility[i]*w1[i+1] + rho**2*(r-q)*dt
180 del w1
181 ST1 = np.exp(logS1)
182 del logS1
183
184 int_var = np.sum(volatility[:-1]**2*dt,axis=0)
185 Q = np.max(int_var)+1e-9
186 del volatility
187 X = (bs.BSCall(ST1, strike_array.reshape(-1,1), T, r, q, np.
    sqrt((1-rho**2)*int_var/T))).T
188 Y = (bs.BSCall(ST1, strike_array.reshape(-1,1), T, r, q, np.
    sqrt(rho**2*(Q-int_var)/T))).T
189 del int_var
190 eY = (bs.BSCall(S0, strike_array.reshape(-1,1), T, r, q, np.
    sqrt(rho**2*Q/T))).T
191
192 c = []
193 for i in range(strike_array.shape[0]):
194     cova = np.cov(X[:,i]+10,Y[:,i]+10)[0,1]
195     varg = np.cov(X[:,i]+10,Y[:,i]+10)[1,1]
196     if (cova or varg)<1e-8:
197         temp = 1e-40
198     else:
199         temp = np.nan_to_num(cova/varg,1e-40)
200     temp = np.minimum(temp,2)
201     c.append(temp)
202 c = np.array(c)
203
204 call_mc_cv1 = X-c*(Y-eY)
205 del X
206 del Y
207 del eY
208
209 return np.average(call_mc_cv1,axis=0)
210
211
212
213 def global_reduction(rho,H,eps,T,a_k_part,S0,strike_array,n_steps,
    N_sims,w1,steps,maturities, index = 0):
214
215     eta_tild = eps**(H-0.5)
216     kappa_tild = (0.5-H)/eps

```

```

217
218 a_0,a_1,a_3,a_5 = a_k_part
219 a_k = np.array([a_0,a_1,0,a_3,0,a_5])
220
221 dt = T/n_steps
222 tt = np.linspace(0., T, n_steps + 1)
223
224 r = iD.r(tt, index)
225 q = iD.q(tt, index)
226
227 exp1 = np.exp(kappa_tild*tt)
228 exp2 = np.exp(2*kappa_tild*tt)
229
230 diff_exp2 = np.concatenate((np.array([0.]),np.diff(exp2)))
231 std_vec = np.sqrt(diff_exp2/(2*kappa_tild))[:,np.newaxis] #to
232 be broadcasted columnwise
233 exp1 = exp1[:,np.newaxis]
234 X = (1/exp1)*(eta_tild*np.cumsum(std_vec*w1, axis = 0))
235 Xt = np.array(X[:-1])
236 del X
237
238 tt = tt[:-1]
239 std_X_t = np.sqrt(eta_tild**2/(2*kappa_tild)*(1-np.exp(-2*
240 kappa_tild*tt)))
241 n = len(a_k)
242
243 cauchy_product = np.convolve(a_k,a_k)
244 normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].reshape
245 (-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
246 scipy.special.factorial2(np.arange(0,2*n,2).reshape(-1,1)
247 -1),axis=0)
248
249 f_func = horner_vector(a_k[:-1], len(a_k), Xt)
250
251 del Xt
252
253 fv_curve = vc.variance_curve(tt, index).reshape(-1,1)
254
255 volatility = f_func/np.sqrt(normal_var.reshape(-1,1))
256 del f_func
257 volatility = np.sqrt(fv_curve)*volatility
258
259 ST1 = list()
260 logS1 = np.log(S0)
261 for i in range(w1.shape[0]-1):
262     logS1 = logS1 - 0.5*dt*(volatility[i]*rho)**2 + np.sqrt(dt)
263     *rho*volatility[i]*w1[i+1] + rho**2*(r[i]-q[i])*dt
264     if i in steps:
265         ST1.append(np.exp(logS1))
266 del w1
267 ST1.append(np.exp(logS1))
268 ST1 = np.array(ST1)
269 del logS1
270
271 int_var = np.sum(volatility[:-1]**2*dt,axis=0)
272 Q = np.max(int_var)+1e-9
273 del volatility

```

```

269
270 P = list()
271
272 for i in range(len(steps)):
273     T_aux = maturities[i]
274     r = iD.r(T_aux, index); q = iD.q(T_aux, index)
275
276     X = (bs.BSCall(ST1[i], strike_array.reshape(-1,1), T_aux, r
277 , q, np.sqrt((1-rho**2)*int_var/T))).T
278     Y = (bs.BSCall(ST1[i], strike_array.reshape(-1,1), T_aux, r
279 , q, np.sqrt(rho**2*(Q-int_var)/T))).T
280     eY = (bs.BSCall(S0, strike_array.reshape(-1,1), T_aux, r, q
281 , np.sqrt(rho**2*Q/T))).T
282
283     c = []
284     for i in range(strike_array.shape[0]):
285         cov = np.cov(X[:,i]+10,Y[:,i]+10)[0,1]
286         var = np.cov(X[:,i]+10,Y[:,i]+10)[1,1]
287         if (cov or var)<1e-8:
288             temp = 1e-40
289         else:
290             temp = np.nan_to_num(cov/var,1e-40)
291             temp = np.minimum(temp,2)
292             c.append(temp)
293     c = np.array(c)
294
295     call_mc_cv1 = X-c*(Y-eY)
296     P.append(np.average(call_mc_cv1,axis=0))
297
298 return np.array(P)

```

# Conclusion