UNIVERSITÁ DEGLI STUDI DI MILANO

FACOLTÁ DI SCIENZE E TECNOLOGIE

Corso di Laurea Magistrale in Matematica

BAYESIAN INFERENCE OF STOCHASTIC VOLATILITY MODELS USED BY MARKET PARTICIPANTS

Relatore:
Prof. Marco Maggis

Correlatore:
Prof. Martin Simon

Tesi di Laurea di:
Lorenzo Proserpio
Matricola 03179A

Anno Accademico 2022-2023

Acknowledgement

Introduction

This thesis aims at addressing the challenges and problems faced by practitioners in the industry. In the exotic equity derivatives and structured products business, there is a need for models both for pricing and hedging the contracts. Quoting George E. P. Box: *"essentially, all models are wrong, but some are useful"*. To be useful, these models need to accurately represent the implied volatility surface while also having a minimal number of parameters that are highly interpretable and can be estimated with precision quickly. Given the existence of numerous models with such properties, it is natural to question which models are being employed by other market participants. The innovative idea of this thesis is to establish a framework which can generate the probability distribution indicating the likelihood that market prices imply the usage of a particular model (on a given set) by other market participants.

A vanilla European option provides the holder with the right to buy (or sell) an asset at a specified price and time in the future. The Black-Scholes-Merton model, presented in [1] and [2] , introduced a theoretical framework that considered various factors such as the underlying asset price, strike price, time to expiration, risk-free interest rate and volatility. Their formula to evaluate options' prices induced the concept of implied volatility surface. The implied volatility surface is a two-dimensional function that relates to tenors and strikes. Each point on the surface is obtained by observing the prices of publicly traded vanilla options and solving the Black-Scholes equation to derive the corresponding implied volatility. According to this model, the implied volatility surface was expected to be flat. However, this did not align with empirical evidences. There was a higher demand for out-the-money options, particularly puts, leading to price increases. This demand had an impact on the implied volatility surface, causing it to deviate from being flat and exhibit a curvature. Heston proposed his own model in [3], which introduced two stochastic equations. One equation was for the stock

price, capturing its dynamics, while the other equation was specifically for modeling the volatility of the underlying asset. In this way, Heston was able to capture and model several stylized facts observed in financial markets. These included the mean-reversion effect, the negative correlation between prices and volatility, the clustering effect, and the presence of skew and term structure. This last properties refer to the asymmetry in implied volatilities across different strike prices and to the variations in implied volatilities across different time-to-expiration periods. However, the skew obtained by using the Heston model did not entirely match what was observed in the market. Thanks to the electronification of the markets and the availability of high-frequency price data, it became possible to gather new empirical evidence indicating that volatility exhibits roughness. The concept of rough volatility has been introduced by Gatheral, Jasson and Rosenbaum in [4]. Fractional stochastic differential equations are now employed in the model. Taking into consideration the stylized effects of the market's microstructure, El Euch, Fukasawa, and Rosenbaum in [5] proposed a model that exhibits a limiting behavior resembling a rough version of the Heston model capable of matching the skew.

The Heston model is widely used due to its capability to capture essential characteristics of low-frequency asset price movements. In addition to theoretical soundness, every model used to price options must address two important questions:

1. Is it easily and quickly calibrable? This pertains to the model's ability to efficiently estimate its parameters based on market data, ensuring a good fit between the model and observed option prices.
2. Do we have a fast numerical scheme to simulate trajectories? This refers to the computational efficiency of generating simulated price trajectories using the model. A fast numerical scheme allows for quick and accurate simulation of asset price paths, which is essential for pricing exotics options and conducting risk analysis.

Addressing these questions is crucial to ensure the practicality and usefulness of an options pricing model in real-world applications. In the Heston model, it is possible to derive a closed-form expression for the conditional characteristic function. This closed-form solution is of great significance in the COS method proposed by Fang and Oosterlee in [6], along with Le Floc'h's correction. The COS method is able to efficiently price vanilla European options with speed and accuracy. We opted to solve the non-linear least squares problem for calibration using the Levenberg-Marquardt algorithm.

This algorithm, in combination with the fast forward map provided by the COS method, enables the calibration of all five parameters of the Heston model in under a minute, even on a standard laptop. To tackle the second question, in Chapter 2 we decided to implement the Gamma Approximation Scheme proposed by Bégin, Bédard and Gaillardetz in [7]. This low-bias scheme is a relatively recent development that demands significantly fewer trajectories in Monte Carlo simulations to achieve the same level of accuracy compared to the Euler or Milstein schemes. In the final part of the chapter, we computed the forward at-the-money skew, which represents the partial derivative of the implied volatility surface with respect to the strike for at-the-money options, both for the market data and the model-generated data. However, we observed a discrepancy between the two. Specifically, the forward at-the-money skew generated by the Heston model appeared to decay at a faster rate compared to the one obtained from the market data.

With the advent of electronic markets, additional stylized effects have been observed to occur at high frequencies. The markets exhibit several characteristics: they are highly endogenous, meaning that market dynamics are influenced by internal factors; they are more efficient at high frequencies; there is asymmetry in the bid and ask sides of the order book; the market is primarily populated by metaorders. Building upon these observed effects, in [5] is developed a model for the market's microstructure employing a bi-dimensional Hawkes process. By taking the limits of these processes over long time horizons, the rough Heston model can be obtained. Even though the Markovian property of the model is indeed lost; nevertheless, we are still able to derive a quasi-closed formula for the conditional characteristic function. The formula is referred to as quasi-closed because it involves a function that serves as the solution to a fractional Riccati differential equation. Gatheral and Radoicic in [8], with the assistance of Alòs, derived a rational approximation for this solution. This approximation, combined with the Lewis Formula, allows for efficient computation of option prices and related quantities. In Chapter 3 we chose to solve the non-linear least squares problem for calibration using the Trust Region Reflective algorithm. This algorithm offers the advantage of allowing us to set parameter bounds, ensuring that the calibrated values remain within specified limits. To simulate the trajectories, we implemented the HQE scheme introduced by Gatheral in [9]. This scheme combines a hybrid step with the QE scheme developed by Andersen in [10]. In the final part of the chapter we showed that the forward at-the-money skew for the model-generated data is matching the market's skew. This alignment can be

attributed to the power law kernel incorporated in the stochastic component of the model, which causes the decay of the skew to be slower compared to the Heston model.

Heston and rough Heston are just two models among the vast selection of options pricing models available. Additionally, we introduced the rough Bergomi model and the Quintic Ornstein-Uhlenbeck model as alternatives. The innovative idea of this thesis is to establish a framework which can generate the probability distribution indicating the likelihood that market prices imply the usage of a particular model (in a given set) by other market participants. The framework developed in Chapter 4 aims at providing valuable insights into the model preferences, which can be useful for traders to understand how other market participants will hedge their positions. This is achieved by employing Approximate Bayesian Computation (ABC), where the traditional summary statistics are replaced with the Sliced Wasserstein distance. The whole process, once you have the calibrated models, can be performed in a couple of minutes on a standard laptop.

Index

Chapter 1

Dataset

The S&P 500 index (ticker: SPX) tracks the performance of 500 large-cap companies listed on major U.S. stock exchanges weighted by market capitalization of the constituent companies. The dataset used included the implied volatility surface for vanilla options of SPX on 23^{rd} January 2023, where moneyness (strike divided by the spot price) ranged from 80% to 120%, and tenors ranged from two weeks to ten years. In the first chapter, we address a question that is frequently overlooked in academic papers, which is how to estimate the drift term. The drift term, defined as the difference between the risk-free rate and the borrowing cost, impacts heavily on the forward value of the underlying. For reference, 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:

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

solving for q we obtain:

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

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:

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

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 $t = 0$:

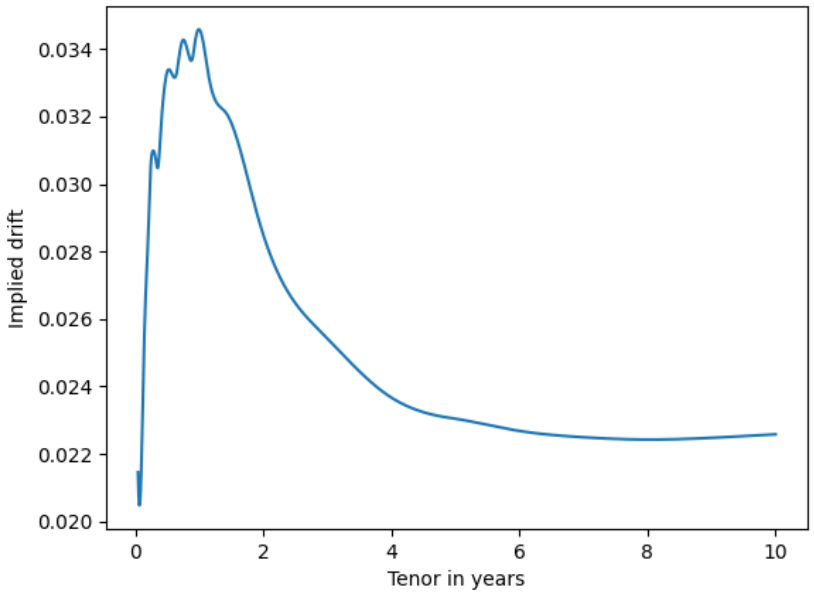
$$\frac{1}{S_0}(C_{0,K_1} - P_{0,K_1} + K_1 e^{-rT}) - \frac{1}{S_0}(C_{0,K_2} - P_{0,K_2} + K_2 e^{-rT})$$

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

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

so in this case the porfolio pays a premium at inception. 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 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.



Chapter 2

Heston

The Heston model, named after its creator Steven Heston, was introduced for the first time in 1993 in [3] 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.

2.1 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 :

$$\begin{cases} \mathrm{d}S_t = \mu S_t \mathrm{d}t + S_t\sqrt{v_t}\mathrm{d}W_t \\ \mathrm{d}v_t = \kappa(\eta - v_t)\mathrm{d}t + \theta\sqrt{v_t}\mathrm{d}\tilde{W}_t \\ v_0 = \sigma_0^2 \end{cases}$$

where:

- μ 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 $\mathrm{d}\langle W, \tilde{W}\rangle_t = \rho \mathrm{d}t$ 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*.

2.2 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 Δ shares of the stock and short Δ_1 contracts of $\tilde{U}$. So the value (denoted with Π_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 sub-

scripts, of the value of the portfolio as:

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

to make our portfolio instantaneously risk-free we must impose the following equations to eliminate the $\mathrm{d}S$ and $\mathrm{d}v$ terms:

$$\begin{cases}
\dfrac{\partial U}{\partial S} - \Delta_1\dfrac{\partial \tilde{U}}{\partial S} - \Delta = 0 \\
\dfrac{\partial U}{\partial v} - \Delta_1\dfrac{\partial \tilde{U}}{\partial v} = 0
\end{cases} \tag{2.1}$$

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:

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

then choosing Δ and Δ_1 as in (2.1) 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):

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

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:

$$\frac{\partial U}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + \rho\theta vS\frac{\partial^2 U}{\partial S\partial v} + \frac{1}{2}\theta^2 v\frac{\partial^2 U}{\partial v^2} + rS\frac{\partial U}{\partial S} - rU = -f\frac{\partial U}{\partial v} \tag{2.2}$$

Equation (2.2) 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 ϕ_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.

2.2.1 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 ϕ. 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 $\mathrm{d}\langle W, W^{\perp}\rangle = 0$. Then we can rewrite the Heston model under $\mathbb{P}$ as:

$$\begin{cases} \mathrm{d}S_t = \mu S_t \mathrm{d}t + S_t\sqrt{v_t}\mathrm{d}W_t \\ \mathrm{d}v_t = \kappa(\eta - v_t)\mathrm{d}t + \theta\rho\sqrt{v_t}\mathrm{d}W_t + \theta\sqrt{1-\rho^2}\sqrt{v_t}\mathrm{d}W_t^{\perp} \end{cases}$$

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

$$\lambda_t \coloneqq \frac{\mu - r}{\sqrt{v_t}}$$

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

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

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:

$$\left.\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\right|_{F_t} \coloneqq Z_t$$

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:

$$W_t^{\mathbb{Q}} = W_t + \int_0^t \lambda_s \mathrm{d}s \text{ and } W_t^{\mathbb{Q},\perp} = W_t + \int_0^t \phi_s \mathrm{d}s$$

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

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

and under this we have that the discounted stock price is a $\mathbb{Q}$ local martingale. However, we can choose any arbitrary ϕ and obtain another $\tilde{\mathbb{Q}}$ with the same properties. But, as in our case, both the claims are actively traded we can infer the ϕ from the market prices of the options, which will determine a $\mathbb{Q}$ and then fix the ϕ. 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 ϕ 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:

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

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

2.3 The Characteristic Formula

2.3.1 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:

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

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

$$-\frac{\partial C}{\partial \tau} + \frac{1}{2}v\frac{\partial^2 C}{\partial x^2} + \rho\theta v\frac{\partial^2 C}{\partial x \partial v} + \frac{1}{2}\theta^2 v\frac{\partial^2 C}{\partial v^2} - \frac{1}{2}v\frac{\partial C}{\partial x} + [\kappa(\eta - v)]\frac{\partial C}{\partial v} = 0 \quad (2.3)$$

according to Duffie, Pan and Singleton in [11], the solution to this equation has the form of:

$$C(x, v, \tau) = K\left\{e^x P_1(x, v, \tau) - P_0(x, v, \tau)\right\} \quad (2.4)$$

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 (2.4) into (2.3) 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:

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

Now take the Fourier transform of P_j as:

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

and the inverse transform given by:

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

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:

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

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:

$$\begin{aligned}f(u,\tau)&=\kappa\left\{r_-\tau-\frac{2}{\theta^2}\log\left(\frac{1-\frac{r_-}{r_+}e^{-d\tau}}{1-\frac{r_-}{r_+}}\right)\right\}\\ g(u,\tau)&=r_-\frac{1-e^{-d\tau}}{1-\frac{r_-}{r_+}e^{-d\tau}}\end{aligned}\tag{2.5}$$

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

$$P_j(x, v, \tau) = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \Re\left\{ \frac{\exp\{f_j(u,\tau)\eta + g_j(u,\tau)v + iux\}}{iu} \right\} \mathrm{d}u \quad (2.6)$$

2.3.2 Numerical integration of the complex logarithm

In (2.5) 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 in [14] proved the following result when using the FFT-like approach:

Proposition 1. *Whenever the parameters of the Heston model are such that* $\Im\{d(u)\} \coloneqq \Im\{\sqrt{(\rho\theta ui - \kappa)^2 + \theta^2(iu + u^2)}\} \neq 0$ *and* $2\kappa\eta \neq \theta^2 n$ *(where* $n \in \mathbb{N}$*), then defining (2.5) 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 (2.5). 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.

2.3.3 Derivation of the characteristic function

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

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

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:

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

then we have that, using Fubini theorem:

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

2.4 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 in [6]. 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.

2.4.1 Inverse Fourier integral via cosine expansion

Remember that:

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

we are interested in approximating:

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

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

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

Now the function $\tilde{f}_H \coloneqq f_H\mathbb{1}_{[a,b]}$ is obviously supported on a finite interval $[a,b]$ and it is continuously differentiable on that interval (look at how it is defined ϕ_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:

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

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 ∞) 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:

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

2.4.2 Pricing european options

Let $x \coloneqq \log\left(\frac{S_0}{K}\right)$ and $y \coloneqq \log\left(\frac{S_T}{K}\right)$ 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:

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

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

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

no we replace $f_H(y|x)$ with the approximation obtained in the previous paragraph:

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

now we define:

$$V_k \coloneqq \frac{2}{b-a}\int_a^b v(y,T)\cos\left(k\pi\frac{y-a}{b-a}\right)\mathrm{d}y$$

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:

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

2.4.3 Coefficients V_k for European put options

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

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

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

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

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 in [16] 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 $z \coloneqq \log\left(\frac{S_T}{S_0}\right)$, then we have that:

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

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

$$\begin{aligned}
V_k &= \frac{2S_0}{b-a}\int_a^b \left[e^{-x}-e^z\right]^+ \cos\left(k\pi\frac{z-a}{b-a}\right)\mathrm{d}z \\
&= \frac{2S_0}{b-a}\int_a^{-x} \left[e^{-x}-e^z\right] \cos\left(k\pi\frac{z-a}{b-a}\right)\mathrm{d}z \\
&= \frac{2S_0e^{-x}}{b-a}\int_a^{-x} \left[1-e^{z+x}\right] \cos\left(k\pi\frac{z-a}{b-a}\right)\mathrm{d}z \\
&= \frac{2S_0e^{-x}}{b-a}\int_a^{-x} \left[1-e^{z+x}\right] \cos\left(k\pi\frac{z-a}{b-a}\right)\mathrm{d}z \\
&= \frac{2}{b-a}\left[K\psi_k(a,-x) - S_0\chi_k(a,-x)\right]
\end{aligned}$$

where:

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

after integration we obtain:

$$\psi_k(a,-x) = \begin{cases} \dfrac{a-b}{k\pi}\sin\left(k\pi\dfrac{x+a}{b-a}\right) & k \neq 0 \\ -x-a & k=0 \end{cases}$$

$$\chi_k(a,-x) = \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2}\left[e^{-x}\cos\left(k\pi\frac{x+a}{b-a}\right) - e^a - \frac{k\pi e^{-x}}{b-a}\sin\left(k\pi\frac{x+a}{b-a}\right)\right]$$

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

2.4.4 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

$$g(u) \coloneqq \log \phi_H(-iu)$$

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:

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

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

2.5 Calibration

In this section we present a method for a fast calibration of the Heston model developed by Cui et al. in [17]. The calibration problem is simply to choose a set of parameters $\Xi \coloneqq [\sigma_0^2, \eta, \rho, \kappa, \theta]^\top$ 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 Ξ. 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:

$$r_i(\Xi) \coloneqq C(\Xi; K_i, T_i) - C^*(K_i, T_i) \qquad i = 1, ..., n$$

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

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

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.

2.5.1 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:

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

then the new form of the ϕ_H is:

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

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 $\mathbf{h}(u) \coloneqq [h_1(u),...,h_5(u)]^\top$ 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:

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

and the last ones:

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

so now we can use the formula with the pseudo-probabilities calculated in the section 2.3.1 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 Ξ:

$$\nabla C(\Xi; K, T) = \frac{e^{-rT}}{\pi}\left[\int_0^\infty \Re\left\{\frac{K^{-iu}}{iu}\nabla\phi_H(\Xi; u-i, T)\right\}\mathrm{d}u - K\int_0^\infty \Re\left\{\frac{K^{-iu}}{iu}\nabla\phi_H(\Xi; u, T)\right\}\mathrm{d}u\right] \quad (2.7)$$

in order to evaluate the two integrals in (2.7) 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 $[0, 100]$.

2.5.2 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:

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

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.

Algorithm 1 Levenberg - Marquardt algorithm

1: Given the initial guess Ξ_0, compute $\|r(\Xi_0)\|$ and J_0.
2: Choose the initial damping factor as $\mu_0 = \tau \max\{diag(J_0)\}$ 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**

In practice works as follow:
As we can see when the iteration is far from the optimum we give a large value to μ_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:

$$\nabla\nabla^\top f \approx JJ^\top$$

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:

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

so whenever the μ_k is sufficiently small we treat the problem as quasi-linear. Notice also that the constraints on parameters such as $\rho \in [-1, 1]$, $\sigma_0 > 0$,

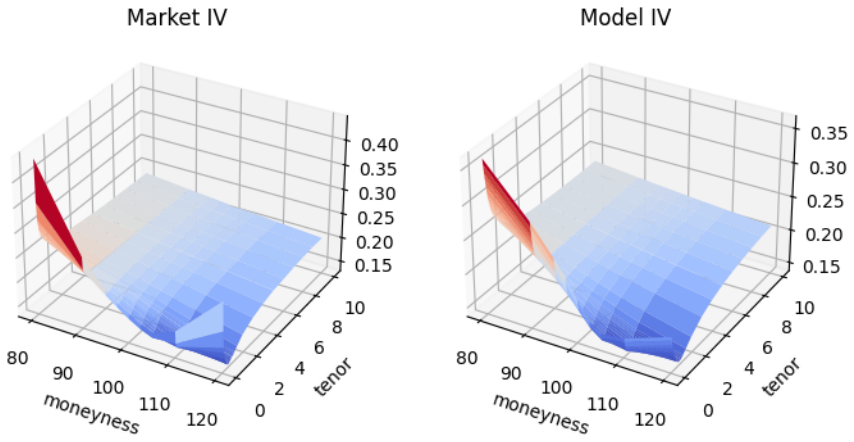
$\kappa > 0$, $\eta > 0$ and $\theta > 0$ are not necessarily satisfied; indeed, whenever we have obtained the output Ξ we need to do a sanity check.

2.5.3 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.

2.6 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 in [18] 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 in [7], 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:

$$\begin{cases} \mathrm{d}X_t = \left(r - q - \frac{1}{2}v_t\right)\mathrm{d}t + \sqrt{v_t}\mathrm{d}W_t \\ \mathrm{d}v_t = \kappa(\eta - v_t)\mathrm{d}t + \theta\sqrt{v_t}\mathrm{d}\tilde{W}_t \\ v_0 = \sigma_0^2 \end{cases}$$

2.6.1 Cumulative distribution function of $v_T|v_t$

We state here an important result, proven in [12], which 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 λ and ν degrees of freedom,*

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

Let

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

and for $t < T$:

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

then we have that

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

2.6.2 Explicit solution for the log-asset price

Let $\Delta t > 0$ be our timestep and discretize time as $0, \Delta t, ..., j\Delta t, ..., T$. Integrating the log-price equation in the Heston model and using the Cholesky decomposition in the Brownian motion we obtain:

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

and in the same way integrating the variance process yields:

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

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

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

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

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

we define the integrated variance as:

$$IV_{(j-1)\Delta t}^{j\Delta t} \coloneqq \int_{(j-1)\Delta t}^{j\Delta t} v_t \mathrm{d}t$$

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

$$\hat{X}_{j\Delta t} = \hat{X}_{(j-1)\Delta t} + (r-q)\Delta t - \frac{1}{2} I\hat{V}_{(j-1)\Delta t}^{j\Delta t} + \frac{\kappa\rho}{\theta} I\hat{V}_{(j-1)\Delta t}^{j\Delta t} + \frac{\rho}{\theta}[\hat{v}_{j\Delta t} - \hat{v}_{(j-1)\Delta t} - \kappa\eta\Delta t] + Z\sqrt{1-\rho^2}\sqrt{I\hat{V}_{(j-1)\Delta t}^{j\Delta t}} \quad (2.8)$$

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

2.6.3 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 2.6.4.
2: Sample $\hat{v}_{j\Delta t}$ given $\hat{v}_{(j-1)\Delta t}$ from the non-central chi-square distribution described in section 2.6.1.
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 (2.8) to obtain $\hat{X}_{j\Delta t}$.

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

2.6.4 Implementation of caches

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

Proposition 4. *The integrated variance admits the representation*

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

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

X_1, X_2 and Z_j have the following representations:

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

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 $\overline{\nu}/2$ and a scale parameter of 1. Finally, ξ is a Bessel random variable with parameter

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

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

Proposition 5. *Let $C_1 = \coth\left[\kappa(T-t)/2\right]$ 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

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

the mean and variance of X_1 respectively satisfy

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

the mean and variance of X_2 respectively satisfy

$$\mathbb{E}[X_2] = \overline{\nu}\theta^2\left[\frac{-2+\kappa(T-t)C_1}{4\kappa^2}\right]$$
$$\mathrm{Var}[X_2] = \overline{\nu}\theta^4\left[\frac{-8+2\kappa(T-t)C_1+\kappa^2(T-t)^2C_2}{8\kappa^4}\right]$$

the mean and the variance of Z respectively satisfy

$$\mathbb{E}[Z] = 4\mathbb{E}[X_2]/\overline{\nu}$$
$$\mathrm{Var}[Z] = 4\,\mathrm{Var}[X_2]/\overline{\nu}$$

Finally,

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

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

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

$$\mathbb{E}[IV^{j\Delta t}_{(j-1)\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] = \mathbb{E}[IV^{*j\Delta t}_{(j-1)\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] + \mathbb{E}[X_1]$$
$$\mathrm{Var}[IV^{j\Delta t}_{(j-1)\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] = \mathrm{Var}[IV^{*j\Delta t}_{(j-1)\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}] + \mathrm{Var}[X_1]$$

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

Notice that $\mathbb{E}[IV^{*j\Delta t}_{(j-1)\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$ and $\mathrm{Var}[IV^{*j\Delta t}_{(j-1)\Delta t}|v_{(j-1)\Delta t}, v_{j\Delta t}]$ 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.

2.7 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 [19] by Gatheral and Jacquier. Without entering too much into the details the model is the following, where k is the forward log-strike,

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

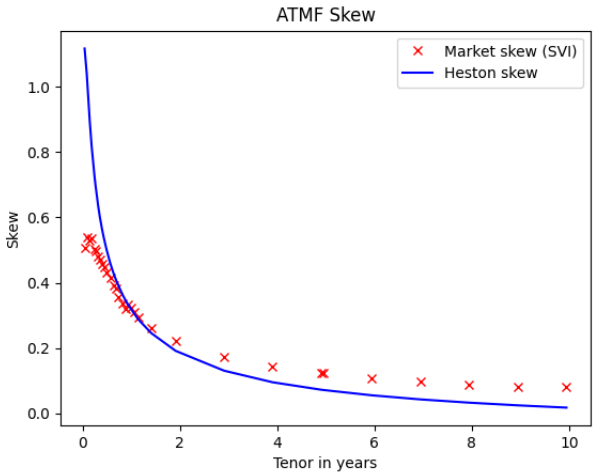
where $a_T \in \mathbb{R}$, $b_T \geq 0$, $|\rho_T| < 1$, $m_T \in \mathbb{R}$ and $\sigma_T > 0$ are parameters different for each tenor. The constraint $a_T + b_T \sigma_T \sqrt{1-\rho_T^2} \geq 0$ guarantees that the implied volatility surface obtained is always positive. Each one of these parameters 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 ψ 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 $T \to \infty$ 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 ATMF 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.

Chapter 3

Rough Heston

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 four main stylized facts that we can observe in market data:

1. Markets are highly endogenous, as showed by Bouchad in [20]. 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 in [21]. 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 to much the state of the market.

In this chapter we will show the rough Heston model, how to calibrate and simulate it efficiently.

3.1 Building the model

As in [5] we will start building a model from Hawkes processes, then slowly including the stylized facts 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.

3.1.1 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 $\mathrm{d}t$ is given by $\lambda_t^+ \mathrm{d}t$, viceversa by λ_t^-. The array $(\lambda_t^+, \lambda_t^-)$ is called intensity of the process and it is of the form:

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

where μ^+ and μ^- 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 λ_t^- can be decomposed in:

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

Now we will see that, when the ϕ_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.

3.1.2 Encoding the 2^{nd} 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:

$$\int_0^t \mathbb{E}[\lambda_s^+]\mathrm{d}s = \mathbb{E}[N_t^+] = \mathbb{E}[N_t^-] = \int_0^t \mathbb{E}[\lambda_s^-]\mathrm{d}s \tag{3.1}$$

remembering how we have defined λ_t^+ and λ_t^-:

$$\mathbb{E}[\lambda_t^+] = \mu^+ + \int_0^t \phi_1(t-s)\,\mathbb{E}[\lambda_s^+]\mathrm{d}s + \int_0^t \phi_3(t-s)\,\mathbb{E}[\lambda_s^-]\mathrm{d}s$$
$$\mathbb{E}[\lambda_t^-] = \mu^- + \int_0^t \phi_4(t-s)\,\mathbb{E}[\lambda_s^-]\mathrm{d}s + \int_0^t \phi_2(t-s)\,\mathbb{E}[\lambda_s^+]\mathrm{d}s$$

the simplest way to satisfy the equation (3.1) is to put:

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

3.1.3 Encoding the 3^{rd} 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 $t \to 0$ we have:

$$\int_0^t \phi_4(t-s)\mathrm{d}N_s^- > \int_0^t \phi_1(t-s)\mathrm{d}N_s^+$$

or equivalently

$$\int_0^t \phi_2(t-s)\mathrm{d}N_s^+ < \int_0^t \phi_3(t-s)\mathrm{d}N_s^-$$

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:

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

3.1.4 Encoding the 1st 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:

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

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:

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

Let $(\Omega, \mathbb{F} = \{(F_T)_{T\geq 0}\}, \mathbb{P})$ a complete filtered probability space. We can find a sequence $\big\{\big(\tilde{\phi}_{1,T}; \tilde{\phi}_{2,T}\big)\big\}_T$ 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:
$$\lim_{T\to\infty}\bigg[\|\tilde{\phi}_{1,T}\|_1 + \beta\|\tilde{\phi}_{2,T}\|_1\bigg] = 1$$

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 Φ. 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 $\left(\tilde{\phi}_{1,T}; \tilde{\phi}_{2,T}\right)$. From now on, we will denote

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

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

3.1.5 Encoding the 4^{th} property

As showed by Jaisson and Rosenbaum in [22], the effect of metaorders are reflected in the Hawkes framework by considering the condition that the kernel matrix exibits 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 $k > 1$, $(\Phi_T)^{*k}(t) = \int_0^t \Phi_T(s)(\Phi_T)^{*(k-1)}(t-s)\mathrm{d}s$. The **Assumption 2** is that Ψ_T is uniformly bounded, Φ is differentiable and the derivative of each component of Φ is bounded and with finite norm 1. In order to satisfy this assumption is sufficient, but not necessary, that ϕ_1 and ϕ_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

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

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

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

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

Wiener-Hopf equations

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

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

given by

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

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

3.1.6 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:

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

then it holds true the following theorem:

Teorema. *As $T \to \infty$ 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:

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

where v_t is the solution to the following rough SDE:

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

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

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

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 [5].

3.1.7 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

$$f^{\alpha,\lambda}(t) = \lambda t^{\alpha-1} E_{\alpha,\alpha}(-\lambda t^\alpha)\mathbf{1}_{t\geq 0}$$
$$F^{\alpha,\lambda}(t) = \int_0^t f^{\alpha,\lambda}(s)\mathrm{d}s$$

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:

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

when not specified β 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*

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

3.1.8 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 section, we can make a very particular choice for the Φ_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 χ is an idempotent matrix, this will be useful later when deriving the characteristic function. Now to identify the appropriate inhomogenous intensity $\hat{\mu}_T(\cdot)$ is far more complicated and not so useful, so we will give only the final candidate:

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

with ε, μ positive constants. In the process to obtain this appropriate candidate, showed by El Euch and Rosenbaum in [25], they obtained also an explicit form for Ψ_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 $T \to \infty$, 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*

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

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

$$v_t = \gamma(t)\varepsilon + \frac{1}{\Gamma(\alpha)}\Bigg[\int_0^t (t-s)^{\alpha-1}\lambda(\gamma(s) - v_s)\mathrm{d}s + \lambda\sqrt{\frac{\gamma(s)(1+\beta^2)}{\lambda\mu(1+\beta)^2}}\int_0^t (t-s)^{\alpha-1}\sqrt{v_s}\mathrm{d}W_s\Bigg]$$

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

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

3.2 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 α with $H + 1/2$. Given a stock price process $S = (S_t)_{t\geq 0}$ the rough Heston model, under risk-free probability measure, is the following :

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

where:

- r is the risk-free rate;
- q is the yield of the underlying;

- W and $\tilde{W}$ are correlated Brownian motions with $\mathrm{d}\langle \tilde{W}, W\rangle_t = \rho \mathrm{d}t$;
- $H \in (0, 1/2)$ is the Hurst exponent of the fractional Brownian motion;
- θ 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

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

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

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

whenever it exists.

3.2.1 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:

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

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

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

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

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

In [24] 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 α, then $\lambda\gamma(\cdot)$ can be chosen so that the model is consistent with the market observed forward variance curve by taking

$$\lambda\gamma(t) = D^\alpha[\xi_0(\cdot) - v_0](t) - \lambda\xi_0(t) \tag{3.2}$$

Equation (3.2) 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 λ is sufficiently small, we can rewrite the dynamic in a compact way as:

$$(\star) \quad \begin{cases} \mathrm{d}S_t = (r-q)S_t\mathrm{d}t + S_t\sqrt{v_t}\left\{\rho\mathrm{d}W_t + \sqrt{1-\rho^2}\mathrm{d}W_t^\perp\right\} \\ v_t = \xi_0(t) + \dfrac{\theta}{\Gamma(H+\frac{1}{2})}\displaystyle\int_0^t \frac{\sqrt{v_s}}{(t-s)^{\frac{1}{2}-H}}\mathrm{d}W_s \end{cases}$$

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

3.3 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:

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

3.3.1 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:

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

then

Teorema 7. *We have*

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

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

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

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 $\Phi : \mathbb{R}^+ \to \mathcal{L}^2(\mathbb{R}^+) \cap \mathcal{L}^1(\mathbb{R}^+)$. 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 $\tau_1^k < \cdots < \tau_{N_t^{0,k}}^k \in [0,t]$ 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 τ_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:

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

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 $X_{(1)}, \cdots, X_{(N^{0,k},t)}$ built from iid variables $X_1, \cdots, X_{N_t^{0,k}}$ with density with support in $[0,t]$. We

will refer to that density as $\frac{\mu_k(s)\mathbf{1}_{s\leq t}}{\int_0^t \mu_k(s)\mathrm{d}s}$ So, if we use the fact that the X_i are iid we obtain:

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

using again independence and rearranging we obtain:

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

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

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

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.

3.3.2 Intuition about the result

Using what we obtained in **section 3.2.8, 3.3.1** and **3.4.1** 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, ξ_0) *is*

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

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

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

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

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

with $\beta > 0$ and we need to use **Theorem 6.**
As written in **section 3.2.8** 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 $T \to \infty$, 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 3.2.8** 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 $\overline{u} \in \mathbb{C}$ and let it be

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

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

$$L^T((\overline{u}_+^T, \overline{u}_-^T), tT) \to L(\overline{u}, t) \quad \text{if} \quad T \to \infty \tag{3.3}$$

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 3.2.8** and the asyntotic properties of the Mittag-Leffler density function given in **section 3.2.7** we may write, for each $t \in (0, 1]$, that

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

Thanks to **Theorem 7.** we can write the characteristic function of our Hawkes process as

$$L^T((\overline{u}_+^T, \overline{u}_-^T), tT) = \exp\Big\{ \int_0^{tT} \hat{\mu}_T(s)\big[C^{T,+}((\overline{u}_+^T, \overline{u}_-^T), tT - s) - 1) + C^{T,-}((\overline{u}_+^T, \overline{u}_-^T), tT - s) - 1)\big]\mathrm{d}s \Big\}$$

where $C^T((\overline{u}_+^T, \overline{u}_-^T), t) = \big(C^{T,+}((\overline{u}_+^T, \overline{u}_-^T), t), C^{T,-}((\overline{u}_+^T, \overline{u}_-^T), t)\big)$ is the solution to its respective integral equation written in **Theorem 7.**. Now we define

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

and we can rewrite the characteristic function as

$$L^T((\overline{u}_+^T, \overline{u}_-^T), tT) = \exp\Big\{ \int_0^{t} T^\alpha\big[\big(Y^{T,+}(\overline{u}, t - s) - 1\big) + \big(Y^{T,-}(\overline{u}, t - s) - 1\big)\big] \cdot \big[T^{1-\alpha}\hat{\mu}(sT)\big]\mathrm{d}s \Big\}$$

Since we have (3) we can expect that as $T \to \infty$ then $T^\alpha\big(Y^T(\overline{u}, t) - (1, 1)\big)$ converges to some functions $\big(c(\overline{u}, t), d(\overline{u}, t)\big)$, 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 $\big(Y^T(\overline{u}, t) - (1, 1)\big) = \mathcal{O}(T^{-\alpha})$ (in the sense component-wise) we can expand $\log\big(Y^T(\overline{u}, t)\big)$ around $(1, 1)$ (where $\log(\cdot)$ has been applied on each component) and obtain

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

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

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

where with $\big(Y^T(\overline{u},t)-(1,1)\big)^2$ we intend $\langle\big(Y^T(\overline{u},t)-(1,1)\big),\big(Y^T(\overline{u},t)-(1,1)\big)\rangle$. Now, we wish to use the Wiener-Hopf lemma to solve the integral part and then using the explicit form for $\Psi_T(T\cdot)$ written in **section 3.2.8**. So we need to notice

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

and then, applying the lemma, we obtain

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

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

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

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

$$\begin{aligned}L(\overline{u},t) &= \exp\big\{\lambda\gamma(t)I^1h(\overline{u},t)+\varepsilon\gamma(t)I^{1-\alpha}h(\overline{u},t)\big\}\\ &= \exp\big\{\lambda\gamma(t)I^1h(\overline{u},t)+v_0I^{1-\alpha}h(\overline{u},t)\big\}\end{aligned}$$

where h is the solution of the fractional Riccati Cauchy problem

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

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.

3.3.3 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 Padè approximants to obtain a fast and reliable approximation. In this section we will follow the work of Gatheral and Radoicic in [8]. 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.

Small times expansion

For the small times expansion we will follow the work of Alòs et al. in [26]. Only for this paragraph suppose that interest rates, borrow costs and yields are zero. Let $H\big(x_t, w_t(T)\big)$ be a solution of this equation

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

where

$$w_t(T) = \mathbb{E}\bigg[\int_t^T v_s \mathrm{d}s \bigg| F_t\bigg] = \int_0^T \xi_0(s)\mathrm{d}s - \int_0^t v_s \mathrm{d}s =: M_t - \int_0^t v_s \mathrm{d}s$$

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*

$$(A \diamond B)_t(T) \coloneqq \mathbb{E}\bigg[\int_t^T d\langle A_t, B_t\rangle_s \bigg| F_t\bigg]$$

provided that the expectation is finite.

Definition 2. *Let $H_t \coloneqq H\big(x_t, w_t(T)\big)$, defined as before, then*

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

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 subscrip 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:

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

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

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

and proceeding with the computations

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

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

$$I_t^{(j)}(T) \coloneqq \int_t^T \xi_t(s)(T-s)^{j\alpha} \mathrm{d}s$$

then we have

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

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:

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

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

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

this clearly satisfy the PDE at the beginning of this section and, moreover, it holds true, trough 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 9.** we have that

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

since we have used **Theorem 9.** 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 $l < j$

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

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

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

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

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

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 θt^α, so if we do the following change of variable $\tilde{t}^\alpha = \theta t^\alpha$ we can rewrite our fractional differential equation as

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

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

Long times expansion

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

Proposition 10. *Let $h_\infty(u,\tilde{t}) \coloneqq r_-[1-E_\alpha(-C\tilde{t}^\alpha)]$. For $u \in \mathbb{C}$ with $\Re(u) \geq 0$ and $\tilde{t} \in \mathbb{R}^+$, if $\tilde{t} \to \infty$ then $h_\infty(u,\tilde{t})$ solves the fractional equation in **Theorem 8.** 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 3.2.7** 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 10.** it raises a natural *ansatz* for $h(u,\tilde{t})$ when $\tilde{t} \to \infty$ 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

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

using the identity principle for power series we obtain

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

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 $t \to \infty$, 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:

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

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

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

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

3.4 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:

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

and then

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

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:

$$T\left(\hat{\sigma}_0^T\right)^2 = \int_0^T \xi_0(s)\mathrm{d}s$$

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:

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

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

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

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.

3.4.1 Fourier transform technique for Vanilla Options

An alternative method has been developed by Carr and Madan in [29] 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

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

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

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

and if $k \to -\infty$, or equivalently $K \to 0$, 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

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

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

$$\begin{aligned}C_{0,K} &= \frac{e^{-\beta k}}{2\pi}\int_{\mathbb{R}} e^{-iuk}\psi(u,T;0)\mathrm{d}u \\ &= \frac{e^{-\beta k}}{\pi}\int_0^\infty e^{-iuk}\psi(u,T;0)\mathrm{d}u \\ &= \frac{e^{-\beta k}}{\pi}\int_0^\infty e^{-iuk}\frac{e^{-rT}\phi_{rH}\big(u-(\beta+1)i,T;0\big)}{\beta^2+\beta-u^2+iu(2\beta+1)}\mathrm{d}u\end{aligned} \tag{3.4}$$

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 $\phi_{rH}\big(-(\beta+1)i,T;0\big)$ being finite and this happens if and only if

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

if we find a suitable β then we have that $\psi \in \mathbb{R}^+$, indeed ϕ_{rH} is bounded by $\mathbb{E}\big[S_T^{\beta+1}\big]$, 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

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

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

3.4.2 Numerical implementation

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

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

where δ_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 λ, so our grid for the strikes is $\big[-b \coloneqq -N\lambda/2, \cdots, N\lambda/2 - \lambda\big] = [k_1, \cdots, k_N]$ so the value at time 0 of a call option with strike K_v with k_v in the grid becomes

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

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

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

where

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

3.5 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 $\Xi \coloneqq [H, \rho, \theta]^\top$ 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 Ξ. Given n call options we define:

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

and the residual vector $r(\Xi) = [r_1(\Xi), ..., r_n(\Xi)]^\top$. 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 8.** 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:

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

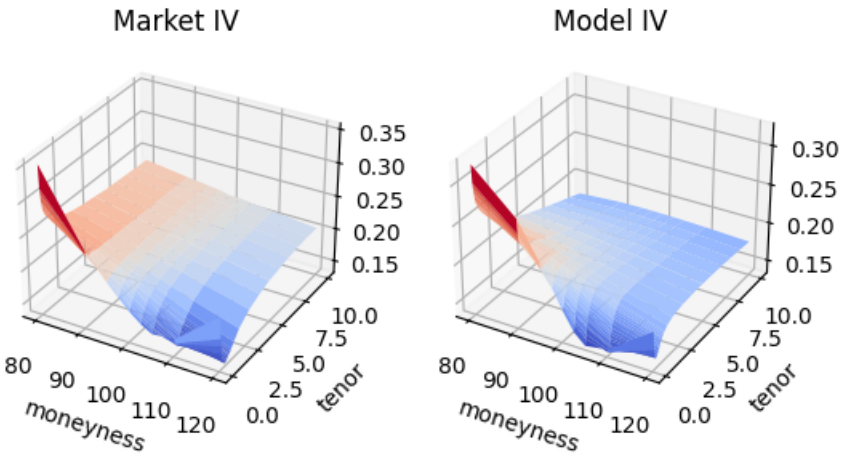
since we suppose to have $n \gg 5$ (where 5 is the number of parameters that we have to determine) it is an overdetermined problem. To tackle this kind of problem we will use the Trust Region Reflective algorithm. For the interested reader see Branch, Coleman, Li in [27].

3.5.1 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 ρ for both the Heston and the rough Heston model.



3.6 Simulation

In this section we will explain the HQE scheme proposed by Gatheral in [9] 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:

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

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

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

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

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

and the martingale increment as

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

so

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

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

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

and, doing the calculations, we obtain

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

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*

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

the last definition that we need is

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

3.6.1 The HQE scheme

Using **Lemma 11.** we have that for $s \in ((n-1)\Delta, n\Delta]$ it is true that $v_s \approx \overline{v}_n$, from there we get the corresponding approximations

$$\mathrm{Var}[\hat{\xi}_{n+1}|F_{n-1}] \approx \overline{v}_n\mathcal{K}_{1,1}$$
$$\mathrm{Var}[\chi_n|F_{n-1}] \approx \overline{v}_n\Delta$$
$$\mathrm{Cov}[u_n, \hat{\xi}_{n+1}|F_{n-1}] \approx \overline{v}_n\mathcal{K}_{0,1}$$
$$\mathrm{Cov}[u_n, \chi_n|F_{n-1}] \approx \overline{v}_n\mathcal{K}_0$$
$$\mathrm{Cov}[\chi_n, \hat{\xi}_{n+1}|F_{n-1}] \approx \overline{v}_n\mathcal{K}_1$$

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 Δ

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

with that our correlation matrix becomes

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

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:

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

Then the algorithm works in this way:
here at each step we simulate only one random variable, Gatheral developed an extension of this for the rough Heston model.

Algorithm 4 QE Scheme

1: Evaluate ψ_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**

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:*

$$\mathbb{E}\left[\frac{\mathcal{K}_0}{\Delta}\tilde{\chi}_n | F_{n-1}\right] = \frac{\hat{\xi}_n}{2}; \quad \mathbb{E}[\tilde{\varepsilon}_n | F_{n-1}] = \frac{\hat{\xi}_n}{2}$$

$$\mathrm{Var}[\tilde{\chi}_n | F_{n-1}] = \overline{v}_n \Delta; \quad \mathrm{Var}[\tilde{\varepsilon}_n | F_{n-1}] = \overline{v}_n \left(\mathcal{K}_{0,0} - \frac{\mathcal{K}_0^2}{\Delta}\right)$$

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

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

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 χ_k, for $k < n$, with ϵ very small, compute

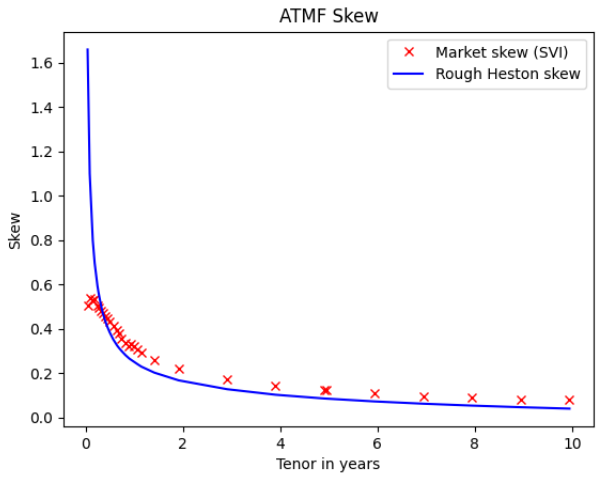
$$\hat{\xi}_n = \max\left[\epsilon, \xi_0(n\Delta) + \sum_{k=1}^{n-1}\sqrt{\frac{\mathcal{K}_{n-k+2,n-k+2}}{\Delta}}\chi_k\right]$$

2: Simulate $\tilde{\chi}_n$ and $\tilde{\varepsilon}_n$ accordingly to **Lemma 12**.
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

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

3.7 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 $T \to \infty$. 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 $T \to \infty$ behaves like $T^{-\alpha} \approx T^{-0.5}$.

Chapter 4

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 [30] by Alessandro Nodari.

4.1 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.

4.1.1 Rough Bergomi model

If we define C_H as

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

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

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

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 $\mathrm{d}\langle \tilde{W}, W\rangle_t = \rho \mathrm{d}t$;
- 2ν 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, ρ, ν) 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.

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 McCricked and Pakkanen in [36]. This is basically a forward Euler scheme paired with an hybrid scheme to simulate the Volterra process

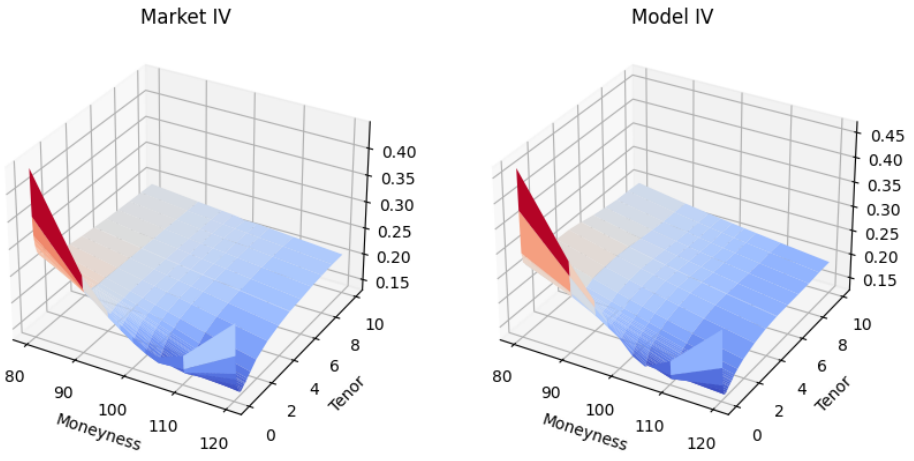
$$\mathcal{V}_t = \int_0^t (t-u)^{H-\frac{1}{2}}\mathrm{d}W_u$$

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.



4.1.2 Quintic Ornstein-Uhlenbeck model

The Quintic Ornstein-Uhlenbeck volatility model, introduced by Jaber, Illand, and Li in [32], 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:

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

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

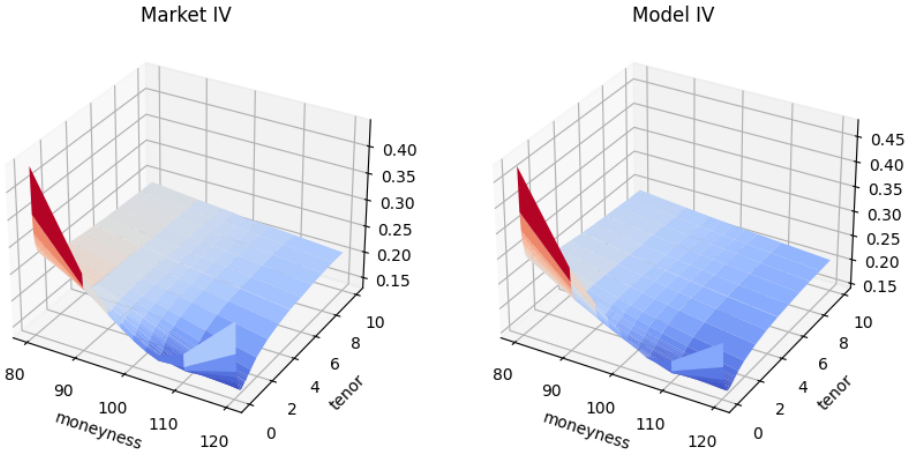
- r is the risk-free rate;
- q is the yield of the underlying;
- W and $\tilde{W}$ are correlated Brownian motions with $\mathrm{d}\langle \tilde{W}, W \rangle_t = \rho \mathrm{d}t$;
- 2ν is the volatility of the volatility;
- $H \in (0, 1/2)$ plays a similar role to the Hurst exponent of the fractional Brownian motion;
- ε 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;
- $p(x) \coloneqq \alpha_0 + \alpha_1 x + \alpha_3 x^3 + \alpha_5 x^5$ 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 α to be non-negative allows the sign of the ATM skew to be the same as ρ. For ε close to zero we have a fast mean-reversion regime and high volatility of volatility, which are two desirable properties to match empirical data.

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:

$$\rho = -0.93924 \quad H = 0.106789 \quad \varepsilon = 0.033779$$
$$\alpha_0 = 0.94018 \quad \alpha_1 = 0.26251 \quad \alpha_3 = 0.064811 \quad \alpha_5 = 0.163232$$

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.

4.2 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 $Y = \{y_j\}_{j=1,...,n}$, 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 θ from a prior distribution $\pi(\cdot)$.
3: Generate synthetic data $Z = \{z_j\}_{j=1,...,n}$ 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** $\theta_1, ..., \theta_N$

it can be proven that if s is a sufficient statistics then, for $\varepsilon \to 0$, 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. in [35] 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.

4.2.1 Sliced Wasserstein distance

The use of a statistics s may lead to a loss of information, in order to avoid that, Bernton et al. in [33] 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

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

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

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

where $\Gamma(x, y)$ is the set of probability measure on $A \times A$ verifying that the marginal distributions are μ and ν 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

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

where σ is the uniform distribution on the unit sphere and $u^*_{\#}\mu$ denote the push-forward measure of μ 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 drawed.

4.3 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

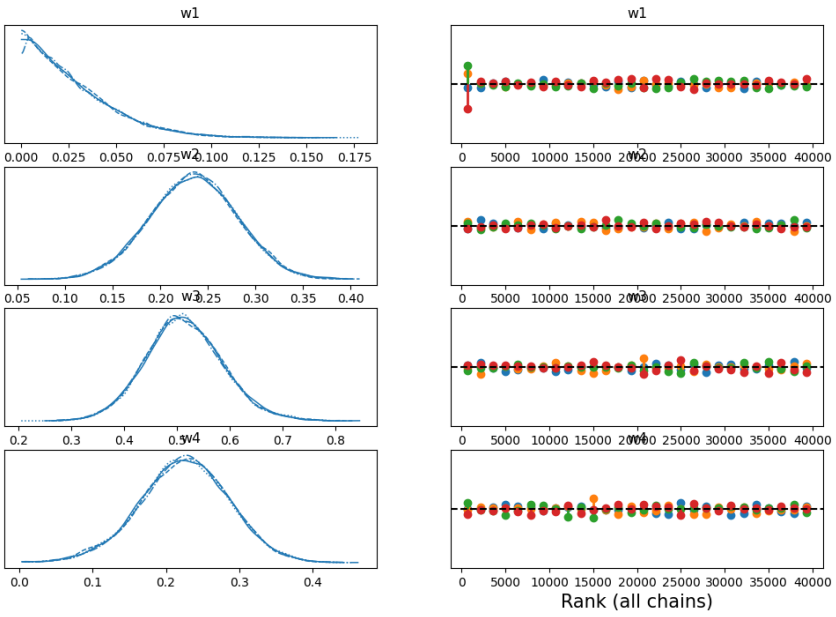
the previous chapter. Our set of models consists in four models: Heston, rough Heston, rough Bergomi and Quintic Ornstein-Uhlenbeck. Each one of them will have a weight, respectively w_1, w_2, w_3 and w_4. We start with uninformative priors, because we do not have any knowledge about which model is prevailing. In order to ensure that the combination is convex we will use:

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

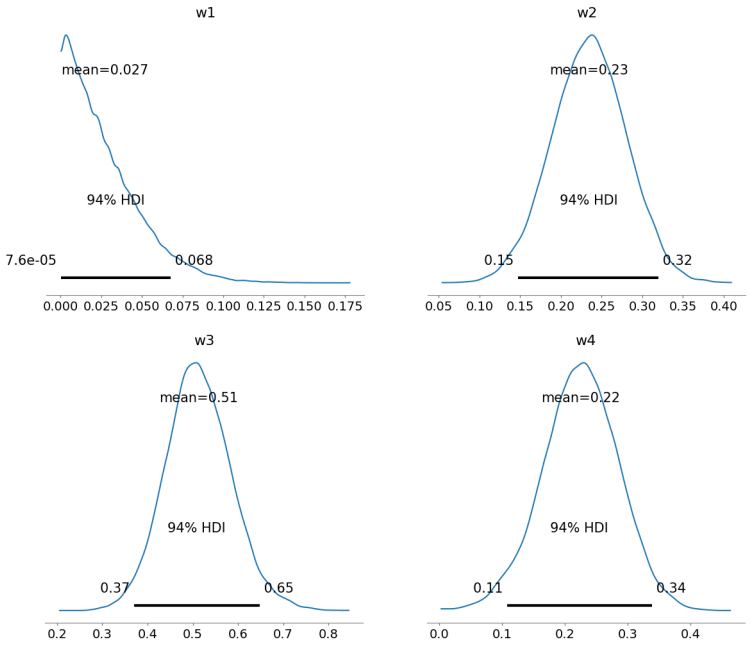
the observations will be the points of the implied volatility surface observed in the market. In this way we will retrieve the distributions for the weights and from there we can assess which models are mostly used. This can be expanded to any arbitrary number of models and, apart from calibrating the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

4.3.1 Numerical results

We used 4 chains with 10000 samples each and $\varepsilon = 0.1$. We obtained the following results:



The figure above 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. These are the 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.

4.3.2 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.

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 κ, η and σ_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 κ, η and σ_0 as priors, while using uninformative priors for θ and ρ. The fair value for a variance swap with tenor T in Heston is:

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

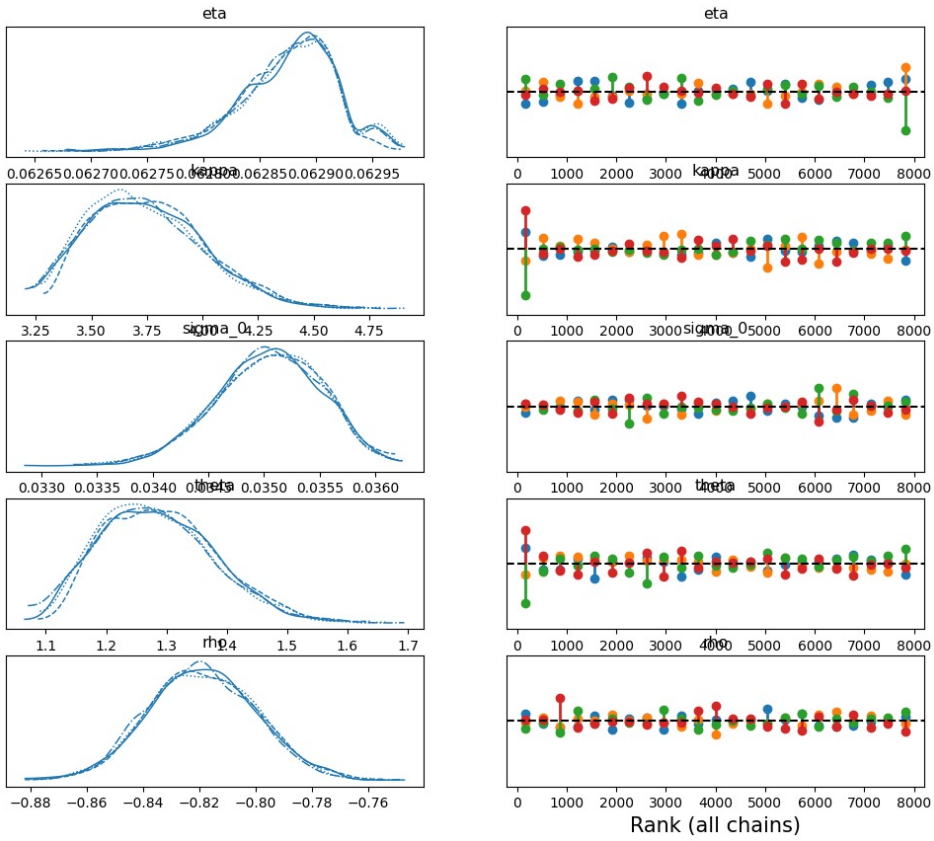
the priors used for the first part were:

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

meanwhile for the second part he used:

- η, κ and σ_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.

Bibliography

[1] R. C. Merton, "Theory of rational option pricing," *Bell J. Econom. and Management Sci.*, vol. 4, pp. 141–183, 1973.

[2] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *J. Polit. Econ.*, vol. 81, no. 3, pp. 637–654, 1973.

[3] S. L. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *Rev. Financ. Stud.*, vol. 6, no. 2, pp. 327–343, 1993.

[4] J. Gatheral, T. Jaisson, and M. Rosenbaum, "Volatility is rough," in *Options—45 years since the publication of the Black-Scholes-Merton model*, vol. 6 of *World Sci. Lect. Notes Finance*, pp. 127–172, World Sci. Publ., Hackensack, NJ, 2023.

[5] O. El Euch, M. Fukasawa, and M. Rosenbaum, "The microstructural foundations of leverage effect and rough volatility," *Finance Stoch.*, vol. 22, no. 2, pp. 241–280, 2018.

[6] F. Fang and C. W. Oosterlee, "A novel pricing method for European options based on Fourier-cosine series expansions," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 826–848, 2008/09.

[7] J.-F. Bégin, M. Bédard, and P. Gaillardetz, "Simulating from the Heston model: a gamma approximation scheme," *Monte Carlo Methods Appl.*, vol. 21, no. 3, pp. 205–231, 2015.

[8] J. Gatheral and R. Radoičić, "Rational approximation of the rough Heston solution," *Int. J. Theor. Appl. Finance*, vol. 22, no. 3, pp. 1950010, 19, 2019.

[9] J. Gatheral, "Efficient simulation of affine forward variance models," *Risk, February*, 2022.

[10] L. B. Andersen, "Efficient simulation of the Heston stochastic volatility model," *SSRN 946405*, 2007.

[11] D. Duffie, J. Pan, and K. Singleton, "Transform analysis and asset pricing for affine jump-diffusions," *Econometrica*, vol. 68, no. 6, pp. 1343–1376, 2000.

[12] D. Dufresne, *The integrated square-root process*. Centre for Actuarial Studies, Department of Economics, University of Melbourne, 2001.

[13] J. Gatheral, *The volatility surface: a practitioner's guide*. John Wiley & Sons, 2011.

[14] H. Albrecher, P. Mayer, W. Schoutens, and J. Tistaert, "The little Heston trap," *Wilmott*, no. 1, pp. 83–92, 2007.

[15] M. Burzoni, *Lecture notes for the course: Mathematical Finance 2*. Università degli Studi di Milano, 2022.

[16] F. L. Floc'h, "More robust pricing of European options based on Fourier cosine series expansions," *arXiv preprint arXiv:2005.13248*, 2020.

[17] Y. Cui, S. del Baño Rollin, and G. Germano, "Full and fast calibration of the Heston stochastic volatility model," *European J. Oper. Res.*, vol. 263, no. 2, pp. 625–638, 2017.

[18] M. Broadie and O. Kaya, "Exact simulation of stochastic volatility and other affine jump diffusion processes," *Oper. Res.*, vol. 54, no. 2, pp. 217–231, 2006.

[19] J. Gatheral and A. Jacquier, "Arbitrage-free SVI volatility surfaces," *Quant. Finance*, vol. 14, no. 1, pp. 59–71, 2014.

[20] J.-P. Bouchaud, "The endogenous dynamics of markets: price impact and feedback loops," *arXiv preprint arXiv:1009.2928*, 2010.

[21] M. K. Brunnermeier and L. H. Pedersen, "Market liquidity and funding liquidity," *The review of financial studies*, vol. 22, no. 6, pp. 2201–2238, 2009.

[22] T. Jaisson and M. Rosenbaum, "Rough fractional diffusions as scaling limits of nearly unstable heavy tailed Hawkes processes," 2016.

[23] O. El Euch, J. Gatheral, and M. Rosenbaum, "Roughening Heston," *Risk*, pp. 84–89, 2019.

[24] O. E. Euch and M. Rosenbaum, "Perfect hedging in rough Heston models," *The Annals of Applied Probability*, vol. 28, no. 6, pp. 3813–3856, 2018.

[25] O. El Euch and M. Rosenbaum, "The characteristic function of rough Heston models," *Math. Finance*, vol. 29, no. 1, pp. 3–38, 2019.

[26] E. Alòs, J. Gatheral, and R. Radoičić, "Exponentiation of conditional expectations under stochastic volatility," *Quant. Finance*, vol. 20, no. 1, pp. 13–27, 2020.

[27] M. A. Branch, T. F. Coleman, and Y. Li, "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems," *SIAM Journal on Scientific Computing*, vol. 21, no. 1, pp. 1–23, 1999.

[28] M. Bennedsen, A. Lunde, and M. S. Pakkanen, "Hybrid scheme for Brownian semistationary processes," *Finance Stoch.*, vol. 21, no. 4, pp. 931–965, 2017.

[29] P. Carr and D. Madan, "Option valuation using the fast Fourier transform," *Journal of computational finance*, vol. 2, no. 4, pp. 61–73, 1999.

[30] A. Nodari, *Bayesian framework to quantify rough volatility calibration risk*. Università degli Studi di Milano, 2023.

[31] P. Piiroinen, L. Roininen, T. Schoden, and M. Simon, "Asset price bubbles: an option-based indicator," *arXiv preprint arXiv:1805.07403*, 2018.

[32] E. A. Jaber, C. Illand, *et al.*, "The quintic Ornstein-Uhlenbeck volatility model that jointly calibrates SPX & VIX smiles," *arXiv preprint arXiv:2212.10917*, 2022.

[33] E. Bernton, P. E. Jacob, M. Gerber, and C. P. Robert, "Approximate Bayesian computation with the Wasserstein distance," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 81, no. 2, pp. 235–269, 2019.

[34] J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder, "Approximate Bayesian computational methods," *Statistics and computing*, vol. 22, no. 6, pp. 1167–1180, 2012.

[35] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde, "Generalized sliced Wasserstein distances," *Advances in neural information processing systems*, vol. 32, 2019.

[36] R. McCrickerd and M. S. Pakkanen, "Turbocharging Monte Carlo pricing for the rough Bergomi model," *Quantitative Finance*, vol. 18, no. 11, pp. 1877–1886, 2018.

[37] L. Bergomi, "Smile Dynamics II," *Risk Magazine*, 2005.

[38] B. B. Mandelbrot and J. W. Van Ness, "Fractional Brownian motions, fractional noises and applications," *SIAM review*, vol. 10, no. 4, pp. 422–437, 1968.

Chapter 5

Libraries

Utils

ImpliedDrift.py

```
import numpy as np
import pandas as pd
from scipy.interpolate import CubicSpline

dates = np.array(["23-01-23","24-01-23","25-01-23","26-01-23",
"27-01-23","30-01-23","06-02-23","13-02-23","21-02-23"])
data = pd.read_csv("ratesOIS.csv")
tenor = np.array(data.TENOR)
Forw = pd.read_csv("forw.csv")
spot = np.array(pd.read_csv("spot.csv").Spot)
TENOR = pd.read_csv("tenor.csv")


def r(x, index = 0):
    rates = np.array(data[dates[index]])/100
    cs = CubicSpline(tenor, rates)
    return cs(x)

def drift(x, index = 0):
    S0 = spot[index]
    F = np.array(Forw[dates[index]]).flatten()
    Tenor = np.array(TENOR[dates[index]]).flatten()
    d = -np.log(S0/F).flatten()/Tenor
    cs = CubicSpline(Tenor, d)
    return cs(x)

def q(x, index = 0):
    return r(x, index) - drift(x, index)
```

BlackScholes.py

```
import numpy as np
from scipy.stats import norm

def BSCall(S0, K, T, r, q, sigma):

    # Price of a call under Black&Scholes

    # S0: spot price
    # K: strike
    # T: years to expiration
    # r: risk free rate (1 = 100%)
    # q: annual yield
    # sigma: volatility (1 = 100%)


    sig = sigma*np.sqrt(T)
    d1 = (np.log(S0/K) + (r-q)*T)/sig + sig/2.
    d2 = d1 - sig

    return S0*norm.cdf(d1) - K*np.exp(-(r-q)*T)*norm.cdf(d2)

def BSPut(S0, K, T, r, q, sigma):

    # Price of a put under Black&Scholes

    # S0: spot price
    # K: strike
    # T: years to expiration
    # r: risk free rate (1 = 100%)
    # q: annual yield
    # sigma: volatility (1 = 100%)

    return BSCall(S0, K, T, r, q, sigma) + K*np.exp(-(r-q)*T) -
     S0

def BSImpliedVol(S0, K, T, r, q, P, Option_type = 1, toll = 1e
    -10):

    # Calculate implied volatility from prices using bisection

    # NOTE: All the parameters can be np.array(), except for P
    that MUST be a np.array().

    # S0: spot price
    # K: strike
    # T: years to expiration
    # r: risk free rate (1 = 100%)
```

```
    # q: annual yield
    # P: prices
    # Option_type: 1 for calls, 0 for puts
    # toll: error in norm 1

    if Option_type:
        BSFormula = np.vectorize(BSCall)
    else:
        BSFormula = np.vectorize(BSPut)

    N = P.shape[0]
    sigma_low = 1e-10*np.ones(N)
    sigma_high = 10*np.ones(N)

    P_low = BSFormula(S0, K, T, r, q, sigma_low)
    P_high = BSFormula(S0, K, T, r, q, sigma_high)
    sigma = (sigma_low + sigma_high)/2.

    while np.sum(P_high - P_low) > toll:
        sigma = (sigma_low + sigma_high)/2.
        P_mean = BSFormula(S0, K, T, r, q, sigma)
        P_low += (P_mean < P)*(P_mean - P_low)
        sigma_low += (P_mean < P)*(sigma - sigma_low)
        P_high += (P_mean >= P)*(P_mean - P_high)
        sigma_high += (P_mean >= P)*(sigma - sigma_high)

    return sigma
```

variance_curve.py

```
import numpy as np

Z1 = np.array([0.23934445564954748, 0.2370172145384514,
    0.23319383855246545, 0.22779527372712297,
    0.22410506177795986, 0.22796530521676028,
    0.22992033119402192, 0.23360387896928214,
    0.23923251959598327])
Z2 = np.array([0.2355916740288041, 0.23547872334450043,
    0.2278527332290963, 0.2493545607795239, 0.2684664268847795,
     0.16758709131144714, 0.23526774429356268,
    0.16354991037070024, 0.09001896821824877])
Z3 = np.array([2.3126258447474375, 2.077549483492911,
    2.1200577204482105, 2.637109433927137, 2.9677374658205307,
    2.385086643216494, 3.064763505035709, 2.3367542064926443,
    2.114562825304444])

# Compute the initial forward variance curve at a given time t
# using the Gompertz function with precomputed parameters

```

```
def Gompertz(t, index = 0):
    z1 = Z1[index]; z2 = Z2[index]; z3 = Z3[index];
    return z1 * np.exp(-z2 * np.exp(-z3 * t))

def variance_curve(t, index = 0):
    z1 = Z1[index]; z2 = Z2[index]; z3 = Z3[index];
    return (z1 * np.exp(-z2 * np.exp(-z3 * t)))**2 + 2*t*z1**2*
    z2*z3*np.exp(-2*z2*np.exp(-z3*t)-z3*t)
```

Heston

Heston.py

```
import numpy as np
import QuantLib as ql
import scipy.integrate
from scipy.special import ive

#################### PUT-CALL PARITY
    ##############################

def put_call_parity(put, S0, strike, r, q, tau):
    # Standard put_call_parity
    return put + S0*np.exp(-q*tau) - strike*np.exp(-r*tau)

def call_put_parity(call, S0, strike, r, q, tau):
    return call - S0*np.exp(-q*tau) + strike*np.exp(-r*tau)



#################### Analytic Heston
    ############################

def phi_hest(u, tau, sigma_0, kappa, eta, theta, rho):

    # Compute the characteristic function for Heston Model

    # u: argument of the function (where you want to evaluate)
    # tau: time to expiration
    # sigma_0, kappa, eta, theta, rho: Heston parameters

    alpha_hat = -0.5 * u * (u + 1j)
    beta = kappa - 1j * u * theta * rho
    gamma = 0.5 * theta ** 2
    d = np.sqrt(beta**2 - 4 * alpha_hat * gamma)
    g = (beta - d) / (beta + d)
    h = np.exp(-d*tau)
    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, kappa, eta, theta, rho)) / \
46                  (u**2 + 0.25))
47
48      i, err = scipy.integrate.quad_vec(integrand, 0, np.inf)
49
50      return i
51
52  def analytic_hest(S0, strikes, tau, r, q,  kappa, theta, rho,
    eta, sigma_0, options_type):
53
54      # Pricing of vanilla options under analytic Heston
55
56      a = np.log(S0/strikes) + (r-q)*tau
57      i = integral(a, tau, sigma_0, kappa, eta, theta, rho)
58
59      out = S0 * np.exp(-q*tau) - strikes * np.exp(-r*tau)/np.pi
    * i
60      out = np.array([out]).flatten()
61
62      for k in range(len(out)):
63          if options_type[k] == 0:
64              out[k] = call_put_parity(out[k], S0, strikes[k], r,
     q, tau)
65
66      return out
67
68  ######################## COS METHOD Le Floch
    ##########################
69
70  def phi_hest_0(u, tau, r, q, sigma_0, kappa, eta, theta, rho):
71
72      # Compute the characteristic function for Heston Model with
     log_asset = 0
73
74      # 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*aux) / (1-g)))
88      term_3 = 1j*(r-q)*u*tau
89
90      return np.exp(term_1)*np.exp(term_2)*np.exp(term_3)
91
92  def chi_k(k, c, d, a, b):
93      # Auxiliary function for U_k
94
95      aux_1 = k*np.pi/(b-a)
96      aux_2 = np.exp(d)
97      aux_3 = np.exp(c)
98
99      return  (np.cos(aux_1*(d-a))*aux_2 - \
100             aux_3 + \
101             aux_1*np.sin(aux_1*(d-a))*aux_2) / (1+aux_1**2)
102
103 def psi_k(k, c, d, a, b):
104     # Auxiliary function for U_k
105
106     if k == 0:
107         return d - c
108
109     aux = k*np.pi/(b-a)
110     return np.sin(aux*(d-a)) / aux
111
112 def U_k_put(k, a, b):
113     # Auxiliary for cos_method
114
115     return 2./(b-a) * (psi_k(k, a, 0, a, b) - chi_k(k, a, 0, a,
        b))
116
117 def optimal_ab(r, q, tau, sigma_0, kappa, eta, theta, rho, L =
    12):
118     # Compute the optimal interval for the truncation
119     aux = np.exp(-kappa* tau)
120     c1 =  (r-q)*tau - sigma_0 * tau / 2
121
```

```
122     c2 = (sigma_0) / (4*kappa**3) * (4*kappa**2*(1+(rho*theta*
    tau-1)*aux) \
123                                      + kappa*(4*rho*theta*(aux
    -1)\
124                                               -2*theta**2*tau*
    aux) \
125                                      +theta**2*(1-aux*aux)) \
126         + eta/(8*kappa**3)*(8*kappa**3*tau - 8*kappa**2*(1+ rho
    *theta*tau +(rho*theta*tau-1)*aux)\
127                             + 2*kappa*((1+2*aux)*theta**2*tau
    +8*(1-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, N):
135     # Auxiliary term precomputed
136
137     a,b = optimal_ab(r, q, tau, sigma_0, kappa, eta, theta, rho
    , L)
138     aux = np.pi/(b-a)
139     out = np.zeros(N-1)
140
141     for k in range(1,N):
142         out[k-1] = np.real(np.exp(-1j*k*a*aux)*\
143                            phi_hest_0(k*aux, tau, r, q, sigma_0,
    kappa, eta, theta, rho))
144
145     return out, a, b
146
147 def V_k_put(k, a, b, S0, K, z):
148     # V_k coefficients for puts
149
150     return 2./(b-a)*(K*psi_k(k, a, z, a, b) - S0*chi_k(k, a, z,
     a, b))
151
152 def cos_method_Heston_LF(precomp_term, a, b, tau, r, q, sigma_0
    , kappa, eta, theta, rho, S0,\
153                            strikes, N, options_type, L=12):
154     # Cosine Fourier Expansion for evaluating vanilla options
    under Heston using LeFloch correction
155     # Should be better for deep otm options.
156
157     # precomp_term: precomputed terms from the function
    precomputed_terms
```

```
158     # a,b: extremes of the interval to approximate
159     # tau: time to expiration (annualized) (must be a number)
160     # r: risk-free-rate
161     # q: yield
162     # sigma_0, kappa, eta, theta, rho: Heston parameters
163     # S0: initial spot price
164     # strikes: np.array of strikes
165     # N: number of terms of the truncated expansion
166     # options_type: binary np.array (1 for calls, 0 for puts)
167     # L: truncation level
168 
169     z = np.log(strikes/S0)
170 
171     out = 0.5 * np.real(phi_hest_0(0, tau, r, q, sigma_0, kappa
        , eta, theta, rho))*\
172           V_k_put(0, a, b, S0, strikes, z)
173 
174     for k in range(1,N):
175         out = out + precomp_term[k-1]*V_k_put(k, a, b, S0,
        strikes, z)
176 
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, tau)
183 
184     return out
185 
186 ########################Calibration
        ######################################
187 
188 def setup_model(_yield_ts, _dividend_ts, _spot,
189                 init_condition):
190     # Setup Heston model object
191 
192     # _yield_ts: Term Structure for yield (QuantLib object)
193     # _dividend_ts: Term Structure for dividend_ts (QuantLib
        object)
194     # init_condition: eta, kappa, theta, rho, sigma_0
195 
196     eta, kappa, theta, rho, sigma_0 = init_condition
197     process = ql.HestonProcess(_yield_ts, _dividend_ts,
198                                ql.QuoteHandle(ql.SimpleQuote(_spot)
        ),
199                                sigma_0, kappa, eta, theta, rho)
200     model = ql.HestonModel(process)
```

```
    engine = ql.AnalyticHestonEngine(model)
    return model, engine

def setup_helpers(engine, expiration_dates, strikes,
                  data, ref_date, spot, yield_ts,
                  dividend_ts, calendar):
    # Helpers for Heston Calibration

    # engine: Heston.setup_model output
    # expiration_dates: maturities
    # data: IV market data
    # ref_date: date for the calculation
    # yield_ts: Term Structure for yield (QuantLib object)
    # dividend_ts: Term Structure for dividend_ts (QuantLib
    object)
    # calendar: type of calendar for calculations

    heston_helpers = []
    grid_data = []
    for i, date in enumerate(expiration_dates):
        for j, s in enumerate(strikes):
            t = (date - ref_date )
            p = ql.Period(t, ql.Days)
            vols = data[i][j]
            helper = ql.HestonModelHelper(
                p, calendar, spot, s,
                ql.QuoteHandle(ql.SimpleQuote(vols)),
                yield_ts, dividend_ts)
            helper.setPricingEngine(engine)
            heston_helpers.append(helper)
            grid_data.append((date, s))
    return heston_helpers, grid_data

def cost_function_generator(model, helpers, norm=False):
    # Define cost function for the calibration (usually Mean
    Square Error)

    def cost_function(params):
        params_ = ql.Array(list(params))
        model.setParams(params_)
        error = [h.calibrationError() for h in helpers]
        if norm:
            return np.sqrt(np.sum(np.abs(error)))
        else:
            return error
    return cost_function

########################Simulation Heston
    ################################
```

```
247
248 def create_totems(base, start, end):
249     # create the grid
250
251     totems = np.ones(end-start+2)
252     index = 1
253     for j in range(start, end+1):
254         totems[index] = base**j
255         index += 1
256
257     totems[0] = 0
258     return totems
259
260 def calc_nu_bar(kappa, eta, theta):
261     # compute v bar
262     return 4*kappa*eta/theta**2
263
264 def x2_exp_var(nu_bar, kappa, theta, dt):
265     # compute E[X_2] and Var[X_2]
266
267     aux = kappa*dt/2.
268     c1 = np.cosh(aux)/np.sinh(aux)
269     c2 = (1./np.sinh(aux))**2
270     exp_x2 = nu_bar*theta**2*((-2.+kappa*dt*c1)/(4*kappa**2))
271     var_x2 = nu_bar*theta**4*((-8.+2*kappa*dt*c1+\
272                                kappa**2*dt**2*c2)/(8*kappa**4))
273     return exp_x2, var_x2
274
275 def Z_exp_var(nu_bar, exp_x2, var_x2):
276     # compute E[Z] and Var[Z]
277
278     return 4*exp_x2/nu_bar, 4*var_x2/nu_bar
279
280 def xi_exp(nu_bar, kappa, theta, dt, totem):
281     # compute E[\Xi] and E[\Xi^2]
282
283     z = 2*kappa*np.sqrt(totem) / (theta**2*np.sinh(kappa*dt/2.)
    )
284     iv_pre = ive(nu_bar/2.-1., z)
285     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         caches_exp[j] = exp_x2 + exp_xi*exp_Z
303         caches_var[j] = var_x2 + exp_xi*var_Z + \
304                         (exp_xi2-exp_xi**2)*exp_Z**2
305 
306     caches_exp[0] = exp_x2
307     caches_var[0] = var_x2
308     return totems, caches_exp, caches_var
309 
310 def x1_exp_var(kappa, theta, dt, vt, vT):
311     # compute E[X_1] and Var[X_1]
312 
313     aux = kappa*dt/2.
314     c1 = np.cosh(aux)/np.sinh(aux)
315     c2 = (1./np.sinh(aux))**2
316 
317     exp_x1 = (vt + vT)*(c1/kappa - dt*c2/2)
318     var_x1 = (vt + vT)*theta**2*(c1/kappa**3 + dt*c2/(2*kappa
    **2) \
319                                  - dt**2*c1*c2/(2*kappa))
320 
321     return exp_x1, var_x1
322 
323 def lin_interp(vtvT, totems, caches_exp, caches_var):
324     # compute linear interpolation for value not in caches
325 
326     exp_int = np.interp(vtvT, totems, caches_exp)
327     var_int = np.interp(vtvT, totems, caches_var)
328     return exp_int, var_int
329 
330 def sample_vT(vt, dt, kappa, theta, nu_bar):
331     # sample vT from a noncentral chisquare (given vt)
332 
333     aux = (theta**2*(1-np.exp(-kappa*dt)))/(4*kappa)
334     n = np.exp(-kappa*dt)/aux *vt
335     return np.random.noncentral_chisquare(nu_bar, n)*aux
336 
337 def generate_path(S0, T, dt, kappa, eta, theta, rho, r, q,
    sigma_0, totems, caches_exp, caches_var):
338     # This function generate one path for the Heston model
    using the Gamma Approx algorithm
339 
```

```
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 + \
377               rho/theta*(vT-vt-kappa*eta*dt) + \
378               z*np.sqrt(1-rho**2)*np.sqrt(iv_t)
379
380         index += 1
381         path[index] = np.exp(xt)
382         vt = vT
383         variance[index] = vt
384         t += dt
385
386     return path[:-1], variance[:-1]
```

Rough Heston

rHeston.py

```
import numpy as np
import ImpliedDrift
import Heston
import BlackScholes
import scipy.integrate

from variance_curve import variance_curve, Gompertz
from scipy.special import gamma
from scipy.interpolate import CubicSpline
from scipy.stats import norm

du = 1e-4
GRID = np.linspace(0,10,int(1./du))


####################### Pade rHeston
    ################################

def Pade33(u, t, H, rho, theta):
    alpha = H + 0.5

    aa = np.sqrt(u * (u + (0+1j)) - rho**2 * u**2)
    rm = -(0+1j) * rho * u - aa
    rp = -(0+1j) * rho * u + aa

    gamma1 = gamma(1+alpha)
    gamma2 = gamma(1+2*alpha)
    gammam1 = gamma(1-alpha)
    gammam2 = gamma(1-2*alpha)

    b1 = -u*(u+1j)/(2 * gamma1)
    b2 = (1-u*1j) * u**2 * rho/(2* gamma2)
    b3 = gamma2/gamma(1+3*alpha) * \
        (u**2*(1j+u)**2/(8*gamma1**2)+(u+1j)*u**3*rho**2/(2*
    gamma2))

    g0 = rm
    g1 = -rm/(aa*gammam1)
    g2 = rm/aa**2/gammam2 * \
         (1 + rm/(2*aa)*gammam2/gammam1**2)

    den = g0**3 +2*b1*g0*g1-b2*g1**2+b1**2*g2+b2*g0*g2

    p1 = b1
```

```
43      p2 = (b1**2*g0**2 + b2*g0**3 + b1**3*g1 + b1*b2*g0*g1 - \
44            b2**2*g1**2 +b1*b3*g1**2 +b2**2*g0*g2 - b1*b3*g0*g2)/
    den
45      q1 = (b1*g0**2 + b1**2*g1 - b2*g0*g1 + b3*g1**2 - b1*b2*g2
    -b3*g0*g2)/den
46      q2 = (b1**2*g0 + b2*g0**2 - b1*b2*g1 - b3*g0*g1 + b2**2*g2
    - b1*b3*g2)/den
47      q3 = (b1**3 + 2*b1*b2*g0 + b3*g0**2 -b2**2*g1 +b1*b3*g1 )/
    den
48      p3 = g0*q3
49
50      y = t**alpha
51
52      return (p1*y + p2*y**2 + p3*y**3)/(1 + q1*y + q2*y**2 + q3*
    y**3)
53
54  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, theta)) / \
172                 (u**2 + 0.25))
173
174     i, err = scipy.integrate.quad_vec(integrand, 0, np.inf)
175
176     return i
177
178 # def integral_vec(x, t, H, rho, theta, grid = GRID):
179 #     aux = (np.tile(grid, (len(x),1)).transpose()*x).transpose
    ()
180
181 #     i = np.real(np.exp(1j*aux)*phi_rhest_vec(grid - 0.5j, t,
    H, rho, theta)) / \
182 #         (grid**2 + 0.25)
183
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
     Lewis Formula
190
191     a = np.log(S0/strikes) + ImpliedDrift.drift(t)*t
192     i = integral(a, t, H, rho, theta)
193     r = ImpliedDrift.r(t)
194     q = ImpliedDrift.q(t)
195     out = S0 * np.exp(-q*t) - np.sqrt(S0*strikes) * np.exp(-(r+
    q)*t*0.5)/np.pi * i
196     out = np.array([out]).flatten()
197
198     for k in range(len(options_type)):
199         if options_type[k] == 0:
200             out[k] = Heston.call_put_parity(out[k], S0, strikes
    [k], r, q, t)
201
202     if (out < 0).any():
203         out[out < 0] = 0.
204
205     return out
206
207 # def analytic_rhest_vec(S0, strikes, t, H, rho, theta,
    options_type):
208
209 #     # Pricing of vanilla options under "analytic" rHeston
    using Lewis Formula
210
```

```
#     a = np.log(S0/strikes) + ImpliedDrift.drift(t)*t
#     i = integral_vec(a, t, H, rho, theta)
#     r = ImpliedDrift.r(t)
#     q = ImpliedDrift.q(t)
#     out = S0 * np.exp(-q*t) - np.sqrt(S0*strikes) * np.exp(-(r+q)*t*0.5)/np.pi * i
#     out = np.array([out]).flatten()

#     for k in range(len(options_type)):
#         if options_type[k] == 0:
#             out[k] = Heston.call_put_parity(out[k], S0, strikes[k], r, q, t)

#     if (out < 0).any():
#         out[out < 0] = 0.

#     return out

######################Simulation rHeston ################################

# Psi for the QE Scheme of Lemma 7.
def psi_m(psi, ev, w):
    #psi minus

    beta2 = psi
    mask = psi > 0
    mask1 = psi <= 0
    if np.any(mask):
        beta2[mask] = 2./psi[mask]-1+np.sqrt(2./psi[mask]* \
                                             np.abs(2./psi[mask]-1))
    if np.any(mask1):
        beta2[mask1] = 0.
    return ev/(1+beta2)*(np.sqrt(np.abs(beta2))+w)**2

def psi_p(psi, ev, u):
    #psi plus

    p = 2/(1+psi)
    res = (u<p)*(-ev)/2*(1+psi)
    mask = u > 0
    if np.any(mask):
        res[mask] = np.log(u[mask]/p[mask])
    return res

# functions for K_i, K_ii and K_01
def Gi(eta, alpha, dt, i):
    return np.sqrt(2*alpha-1)*eta/alpha * dt**alpha * ((i+1)**
```

```
    alpha - i**alpha)

def Gii(eta, H, dt, i):
    aux = 2*H
    return eta**2 * dt**aux * ((i+1)**aux - i**aux)

def G01(eta, alpha, dt):
    return Gi(eta,alpha,dt,0)*Gi(eta,alpha,dt,1)/dt

def HQE_sim(theta, H, rho, T, S0, paths, steps, eps0 = 1e-10):
    # HQE scheme

    # theta, H, rho: parameters of the rHeston model
    # T: final time of the simulations, in years
    # S0: spot price at time 0
    # paths: number of paths to simulate
    # steps: number of timesteps between 0 and T
    # eps0: lower bound for xihat

    dt = T/steps
    dt_sqrt = np.sqrt(dt)
    alpha = H + 0.5
    eta = theta/(gamma(alpha)*np.sqrt(2*H))
    rho2m1 = np.sqrt(1-rho*rho)

    W = np.random.normal(0.,1.,size = (steps,paths))
    Wperp = np.random.normal(0.,1.,size = (steps,paths))
    Z = np.random.normal(0.,1.,size = (steps,paths))
    U = np.random.uniform(0.,1.,size = (steps,paths))
    Uperp = np.random.uniform(0.,1.,size = (steps,paths))

    tj = np.arange(0,steps,1)*dt
    tj += dt

    xij = variance_curve(tj)
    G0del = Gi(eta,alpha,dt,0)
    G00del = Gii(eta,alpha,dt,0)
    G11del = Gii(eta,alpha,dt,1)
    G01del = G01(eta,alpha,dt)
    G00j = np.zeros(steps)

    for j in range(steps):
        G00j[j] = Gii(eta,H,dt,j)
    bstar = np.sqrt((G00j)/dt)

    rho_vchi = G0del/np.sqrt(G00del*dt)
    beta_vchi = G0del/dt

    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
             # Maturity
```

```
14          self.n = n
              # Steps per year
15          self.dt = 1.0/self.n
              # Step size
16          self.s = np.round(self.n * self.T).astype(int)
              # Number of total steps
17          self.t = np.linspace(0, self.T, 1 + self.s)[np.newaxis
    ,:]     # Time grid
18          self.a = a
              # Alpha
19          self.N = N
              # Number of paths
20
21          # Construct hybrid scheme correlation structure with
    kappa = 1
22          self.e = np.array([0,0])
23          self.c = cov(self.a, self.n)
24
25      def dW1(self):
26          np.random.seed(0)
27          # Produces random numbers for variance process with
    required covariance structure
28          N = int(self.N/2)
29          w = np.random.multivariate_normal(self.e, self.c, (N,
    self.s))
30          return np.concatenate((w,-w), axis = 0)
31
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 2d Brownian increments
41
42          Y1 = np.zeros((self.N, 1 + self.s)) # Exact integrals
43          Y2 = np.zeros((self.N, 1 + self.s)) # Riemann sums
44
45          Y1[:,1 : self.s+1] = dW[:, :self.s, 1]   # Assumes
    kappa = 1
46
47          # Construct arrays for convolution
48          G = np.zeros(1 + self.s) # Gamma
49          for k in np.arange(2, 1 + self.s, 1):
50              G[k] = g(b(k, self.a)/self.n, self.a)
51
```

```
52          X = dW[:,:,0] # Xi
53
54          # Compute convolution and extract relevant terms
55          for i in range(self.N):
56              Y2[i,:] = np.convolve(G, X[i,:])[:1+self.s]
57
58          # Finally contruct and return full process
59          return np.sqrt(2 * self.a + 1) * (Y1 + Y2)
60
61      def dZ(self, dW1, dW2, rho):
62          # Constructs correlated price Brownian increments, dB
63
64          self.rho = rho
65          return rho * dW1[:,:,0] + np.sqrt(1 - rho**2) * dW2
66
67      def V(self, Y, xi, eta):
68          # rBergomi variance process.
69          self.xi = xi
70          self.eta = eta
71          a = self.a
72          t = self.t
73          return xi * np.exp(eta * Y - 0.5 * eta**2 * t**(2 * a +
     1))
74          #return xi * ne.evaluate('exp(eta * Y - 0.5 * eta**2 *
    t**(2 * a + 1))')
75
76      def S_all_path(self, V, dZ, r, q, S0):
77          # rBergomi price process.
78          self.S0 = S0
79          dt = self.dt
80          rho = self.rho
81
82          # Construct non-anticipative Riemann increments
83          increments = np.sqrt(V[:,:-1]) * dZ - 0.5 * V[:,:-1] *
    dt + (r - q) * dt
84          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
```

```
        exponent = np.zeros(self.N)
        for i in range(self.s):
            exponent += np.sqrt(V[:,i]) * dZ[:,i] - 0.5 * V[:,i
    ] * dt + (r - q) * dt

        return S0 * np.exp(exponent)

    def global_S(self, V, dZ, S0, steps, index = 0):
        # rBergomi price process.
        self.S0 = S0
        dt = self.dt
        rho = self.rho

        r = iD.r(self.t[0], index)
        q = iD.q(self.t[0], index)

        S = list()
        logS = np.log(S0)
        for i in range(self.s):
            logS += np.sqrt(V[:,i]) * dZ[:,i] - 0.5 * V[:,i] *
    dt + (r[i] - q[i]) * dt
            if i in steps:
                S.append(np.exp(logS))

        S.append(np.exp(logS))

        return np.array(S)
```

Quintic Ornstein-Uhlenbeck

```
import numpy as np
import variance_curve as vc
import ImpliedDrift as iD
import scipy
import BlackScholes as bs

from scipy.integrate import quad

def horner_vector(poly, n, x):
    #Initialize result
    result = poly[0].reshape(-1,1)
    for i in range(1,n):
        result = result*x + poly[i].reshape(-1,1)
    return result



def gauss_dens(mu,sigma,x):
```

```
19      return 1/np.sqrt(2*np.pi*sigma**2)*np.exp(-(x-mu)**2/(2*
        sigma**2))
20
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      delt = 30/365
33      T_delta = T + delt
34
35      std_X = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
        kappa_tild*T)))
36      dt = delt/(n_steps)
37      tt = np.linspace(T, T_delta, n_steps+1)
38
39      FV_curve_all_vix = vc.variance_curve(tt, index)
40
41      exp_det = np.exp(-kappa_tild*(tt-T))
42      cauchy_product = np.convolve(a_k,a_k)
43
44      std_Gs_T = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
        kappa_tild*(tt-T))))
45      std_X_t = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
        kappa_tild*tt)))
46      std_X_T = std_X
47
48      n = len(a_k)
49
50      normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].
        reshape(-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
51      scipy.special.factorial2(np.arange(0,2*n,2).reshape(-1,1)
        -1),axis=0) #g(u)
52
53      beta = []
54      for i in range(0,2*n-1):
55          k_array = np.arange(i,2*n-1)
56          beta_temp = ((std_Gs_T**((k_array-i).reshape(-1,1))*((
        k_array-i-1)%2).reshape(-1,1)*\
57                  scipy.special.factorial2(k_array-i-1).reshape(-1,1)
        *\
58                  (scipy.special.comb(k_array,i)).reshape(-1,1))*\
```

```
            exp_det**(i))*cauchy_product[k_array].reshape(-1,1)
        beta.append(np.sum(beta_temp,axis=0))

    beta = np.array(beta)*FV_curve_all_vix/normal_var
    beta = (np.sum((beta[:,:-1]+beta[:,1:])/2,axis=1))*dt

    sigma = np.sqrt(eps**(2*H)/(1-2*H)*(1-np.exp((2*H-1)*T/eps)
    ))

    f = lambda x: np.sqrt(horner_vector(beta[::-1], len(beta),
    x)/delt)*100 * gauss_dens(0, sigma, x)

    Ft, err = quad(f, -np.inf, np.inf)

    return Ft * np.exp((r-q)*T)



def vix_iv(H, eps, T, a_k_part, K, r, q, n_steps, index = 0):

    a2,a4 = (0,0)
    a0,a1,a3,a5 = a_k_part
    a_k = np.array([a0, a1, a2, a3, a4, a5])

    kappa_tild = (0.5-H)/eps
    eta_tild = eps**(H-0.5)

    delt = 30/365
    T_delta = T + delt

    std_X = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
    kappa_tild*T)))
    dt = delt/(n_steps)
    tt = np.linspace(T, T_delta, n_steps+1)

    FV_curve_all_vix = vc.variance_curve(tt, index)

    exp_det = np.exp(-kappa_tild*(tt-T))
    cauchy_product = np.convolve(a_k,a_k)

    std_Gs_T = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
    kappa_tild*(tt-T))))
    std_X_t = eta_tild*np.sqrt(1/(2*kappa_tild)*(1-np.exp(-2*
    kappa_tild*tt)))
    std_X_T = std_X

    n = len(a_k)

    normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].
```

```
    reshape(-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
103     scipy.special.factorial2(np.arange(0,2*n,2).reshape(-1,1)
    -1),axis=0) #g(u)
104
105     beta = []
106     for i in range(0,2*n-1):
107         k_array = np.arange(i,2*n-1)
108         beta_temp = ((std_Gs_T**((k_array-i).reshape(-1,1))*((
    k_array-i-1)%2).reshape(-1,1)*\
109                 scipy.special.factorial2(k_array-i-1).reshape(-1,1)
    *\
110                 (scipy.special.comb(k_array,i)).reshape(-1,1))*\
111                 exp_det**(i))*cauchy_product[k_array].reshape(-1,1)
112         beta.append(np.sum(beta_temp,axis=0))
113
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], len(beta), x)/delt)*100 - K[i], 0) * gauss_dens(0,
    sigma, x)
124         P[i], err = quad(f, -np.inf, np.inf)
125
126     return P * np.exp((r-q)*T)
127
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,N_sims,w1,r,q, index = 0):
139
140     eta_tild = eps**(H-0.5)
141     kappa_tild = (0.5-H)/eps
142
143     a_0,a_1,a_3,a_5 = a_k_part
```

```
144     a_k = np.array([a_0,a_1,0,a_3,0,a_5])
145 
146     dt = T/n_steps
147     tt = np.linspace(0., T, n_steps + 1)
148 
149     exp1 = np.exp(kappa_tild*tt)
150     exp2 = np.exp(2*kappa_tild*tt)
151 
152     diff_exp2 = np.concatenate((np.array([0.]),np.diff(exp2)))
153     std_vec = np.sqrt(diff_exp2/(2*kappa_tild))[:,np.newaxis] #
    to be broadcasted columnwise
154     exp1 = exp1[:,np.newaxis]
155     X = (1/exp1)*(eta_tild*np.cumsum(std_vec*w1, axis = 0))
156     Xt = np.array(X[:-1])
157     del X
158 
159     tt = tt[:-1]
160     std_X_t = np.sqrt(eta_tild**2/(2*kappa_tild)*(1-np.exp(-2*
    kappa_tild*tt)))
161     n = len(a_k)
162 
163     cauchy_product = np.convolve(a_k,a_k)
164     normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].
    reshape(-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
165         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 be broadcasted columnwise
```

```
    exp1 = exp1[:,np.newaxis]
    X = (1/exp1)*(eta_tild*np.cumsum(std_vec*w1, axis = 0))
    Xt = np.array(X[:-1])
    del X

    tt = tt[:-1]
    std_X_t = np.sqrt(eta_tild**2/(2*kappa_tild)*(1-np.exp(-2*
kappa_tild*tt)))
    n = len(a_k)

    cauchy_product = np.convolve(a_k,a_k)
    normal_var = np.sum(cauchy_product[np.arange(0,2*n,2)].
reshape(-1,1)*std_X_t**(np.arange(0,2*n,2).reshape(-1,1))*\
        scipy.special.factorial2(np.arange(0,2*n,2).reshape
(-1,1)-1),axis=0)

    f_func = horner_vector(a_k[::-1], len(a_k), Xt)

    del Xt

    fv_curve = vc.variance_curve(tt, index).reshape(-1,1)

    volatility = f_func/np.sqrt(normal_var.reshape(-1,1))
    del f_func
    volatility = np.sqrt(fv_curve)*volatility

    ST1 = list()
    logS1 = np.log(S0)
    for i in range(w1.shape[0]-1):
        logS1 = logS1 - 0.5*dt*(volatility[i]*rho)**2 + np.sqrt
(dt)*rho*volatility[i]*w1[i+1] + rho**2*(r[i]-q[i])*dt
        if i in steps:
            ST1.append(np.exp(logS1))
    del w1
    ST1.append(np.exp(logS1))
    ST1 = np.array(ST1)
    del logS1

    int_var = np.sum(volatility[:-1,]**2*dt,axis=0)
    Q = np.max(int_var)+1e-9
    del volatility

    P = list()

    for i in range(len(steps)):
        T_aux = maturities[i]
        r = iD.r(T_aux, index); q = iD.q(T_aux, index)

        X = (bs.BSCall(ST1[i], strike_array.reshape(-1,1),
```

```
    T_aux, r, q, np.sqrt((1-rho**2)*int_var/T))).T
277         Y = (bs.BSCall(ST1[i], strike_array.reshape(-1,1),
    T_aux, r, q, np.sqrt(rho**2*(Q-int_var)/T))).T
278         eY = (bs.BSCall(S0, strike_array.reshape(-1,1), T_aux,
    r, q, np.sqrt(rho**2*Q/T))).T
279
280         c = []
281         for i in range(strike_array.shape[0]):
282             cova = np.cov(X[:,i]+10,Y[:,i]+10)[0,1]
283             varg = np.cov(X[:,i]+10,Y[:,i]+10)[1,1]
284             if (cova or varg)<1e-8:
285                 temp = 1e-40
286             else:
287                 temp = np.nan_to_num(cova/varg,1e-40)
288             temp = np.minimum(temp,2)
289             c.append(temp)
290         c = np.array(c)
291
292         call_mc_cv1 = X-c*(Y-eY)
293         P.append(np.average(call_mc_cv1,axis=0))
294
295     return np.array(P)
```