Quantitative Finance

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/rquf20

Deep calibration with random grids

Fabio Baschetti, Giacomo Bormetti & Pietro Rossi

To cite this article: Fabio Baschetti, Giacomo Bormetti & Pietro Rossi (2024) Deep calibration with random grids, Quantitative Finance, 24:9, 1263-1285, DOI: 10.1080/14697688.2024.2332375

To link to this article: https://doi.org/10.1080/14697688.2024.2332375

Published online: 12 Apr 2024.

Quantitative Finance, 2024
Vol. 24, No. 9, 1263–1285, https://doi.org/10.1080/14697688.2024.2332375

Deep calibration with random grids

FABIO BASCHETTI †, GIACOMO BORMETTI ‡* and PIETRO ROSSI §¶

†Scuola Normale Superiore, Pisa, Italy
‡Department of Mathematics, University of Bologna, Bologna, Italy
§Prometeia S.p.A., Bologna, Italy
¶Department of Statistical Sciences, University of Bologna, Bologna, Italy

(*Received 30 August 2023; accepted 7 March 2024; published online 12 April 2024*)

We propose a neural network-based approach to calibrating stochastic volatility models, which combines the pioneering grid approach by Horvath et al. [Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models. *Quant. Finance,* 2021, **21**(1), 11–27]. with the pointwise two-stage calibration of Bayer and Stemper [Deep calibration of rough stochastic volatility models. Working Paper, arXiv:1810.03399, 2018] and Liu et al. [A neural network-based framework for financial model calibration. *J. Math. Ind.*, 2019, **9**(1), 1–28]. Our methodology inherits robustness from the former while not suffering from the need for interpolation/extrapolation techniques, a clear advantage ensured by the pointwise approach. The crucial point to the entire procedure is the generation of implied volatility surfaces on random grids, which one dispenses to the network in the training phase. We support the validity of our calibration technique with several empirical and Monte Carlo experiments for the rough Bergomi and Heston models under a simple but effective parametrization of the forward variance curve. The approach paves the way for valuable applications in financial engineering—for instance, pricing under local stochastic volatility models—and extensions to the fast-growing field of path-dependent volatility models.

Keywords: Neural network pricing and calibration; Rough volatilty; Forward variance curve

1. Introduction

Calibration of stochastic volatility models is a long-standing problem in quantitative finance. From a theoretical viewpoint, the issue is well understood and requires the implementation of standard optimization routines. However, some operational difficulties often arise regarding the loop over the pricing function. Recent models in the rough regime—such as rough Bergomi (rBergomi) of Bayer *et al.* (2016), quadratic rough Heston (rHeston) of Gatheral *et al.* (2020), and evolutions thereof—typically do not provide a (semi) closed-form expression for the characteristic function (CF) of the asset log-price, the only exception being the rHeston model of Euch and Rosenbaum (2018) (see also El Euch and Rosenbaum 2019). On the contrary, these models—not to mention path-dependent volatility (PDV) models (Guyon 2014, Blanc *et al.* 2017, Gatheral *et al.* 2020, Guyon and Lekeufack 2023, Parent 2023) – are purely simulative and only treated via Monte Carlo techniques which make calibration prohibitive (if not impossible) in terms of computational efforts. The same rHeston model, though amenable to treatment using Fast Fourier Transform (FFT) techniques, is more complex to calibrate than its classical counterpart. The CF requires numerical approximations that slow calibration down (see Diethelm *et al.* 2004, Gatheral and Radoicic 2019, Callegaro *et al.* 2021). It is not surprising, then, that calibration of rough volatility models is now usually dealt with using neural networks. Of course, nothing prevents the application of neural networks to standard stochastic volatility models, and one can readily find several examples in the literature. Using neural networks to calibrate financial models dates back to Hernandez (2017), which deals with the Hull-White model.

Hernandez (2017) approximates the map from implied volatilities to model parameters via a large feed-forward neural network. The author calibrates volatility surfaces – normalized w.r.t. strike and maturity—to associate each with their optimal parameters and trains the network to learn the relationship between volatilities and parameter values. In this direct approach, one feeds newly coming normalized volatility surfaces to the neural network and recovers optimal parameters as an output. Bayer and Stemper (2018) represents the first attempt of a two-stage calibration process. A million combinations of rBergomi parameters are drawn uniformly

*Corresponding author. Email: giacomo.bormetti@unibo.it

and randomly from the parameter space. Each is associated with a strike-expiration pair, and the corresponding option contract is valued using Monte Carlo methods. The authors use the synthetic dataset for training a large neural network (3 hidden layers of 4096 nodes each) to learn the pricing function from model parameters to implied volatilities. Offline training corresponds to step one in the two-stage process. The learned map is stored as neural network weights and carried to the second step, corresponding to the proper calibration. The idea is to replace the 'true' pricing function with the neural network approximation and exploit the numerical advantages offered by the neural networks, such as the fast computation by automatic differentiation of the derivatives to set up gradient-based optimization. In Liu *et al.* (2019), the authors apply the same approach of Bayer and Stemper (2018) to Heston and Bates stochastic volatility models. The CF is available in closed form for these models, and in the generation phase, Fourier integration replaces Monte Carlo pricing. The network is still very large (4 hidden layers of 200 nodes each), and the number of generated samples is again one million.

The entire procedure in Bayer and Stemper (2018) and Liu *et al.* (2019) goes under the name of the *pointwise approach*: The network outputs individual points in implied volatility corresponding to a given strike and maturity. Henceforth, we will enforce the definition of a truly pointwise approach to underline that the generation phase guarantees that each sampled parameter set is associated with a unique option price. We will contrast our approach with the standard *grid-based approach* by Horvath *et al.* (2021). The latter employs (deep) neural networks to approximate the underlying pricing function from model parameters to volatility grids. The idea is to look at the volatility surface as a collection of pixels over a bi-dimensional grid in strike and time to maturity. Training can be performed on a (relatively small) number of couples $(\boldsymbol{\theta}, \sigma_{BS}(K^-, T^-))$, where $\boldsymbol{\theta}$ is one random set of model parameters and $\sigma_{BS}(\cdot,\cdot)$ are the associated implied volatilities over the specified grid (K^-, T^-). As before, this methodology is a two-step approach, where sample generation and training correspond to step one, and calibration represents step two. While one can perform data generation and training offline, calibration is extremely fast and well-suited for online execution. The optimizer replaces each call to the 'true' pricing function by its neural network approximation $F^{\mathcal{M}}(\boldsymbol{\theta}, \cdot)$ and exploits the availability of the gradient to ensure convergence to the minimum in a few milliseconds. Indeed, every network evaluation yields one whole grid, and calibration requires only a few iterations. Two points are worth noting about the grid-based approach in Horvath *et al.* (2021): (i) generation of the samples is much faster, with 80 000 parameters combinations only (each of which is associated with 88 implied volatilities corresponding to an 11×8 strike-time to maturity grid) for the rBergomi model; (ii) the neural network is very small (4 hidden layers of 30 nodes each) when compared with Bayer and Stemper (2018) for the rBergomi model and with Liu *et al.* (2019) for classic stochastic volatility models.

Finally, Rømer (2022) enhances the grid-based approach. The grid is much larger, including very short time to maturities, and the strike specification is conditional on the time to maturity. Such a choice of the grid points is more financially sensible and mitigates some need for more flexibility coming with the approach by Horvath *et al.* (2021). Also, it addresses the need for higher accuracy in the wings of the smile. The author selects 64 maturities with 25 strikes each and trains six different networks to keep sizes low (3 hidden layers, 200 nodes per layer), acknowledging that 'short- and long-term volatility smiles behave rather differently, especially under rough volatility'. Numerical experiments deal with classic and rough stochastic volatility models and investigate the joint calibration to SPX and VIX smiles.

The key to every two-step approach—both on a grid or pointwise—is that the neural network is much faster to evaluate than the actual pricing function it is approximating. A significant difference, however, is immediately apparent as the grid-based approach requires interpolation/extrapolation steps that the pointwise method gets rid of. In fact, in a grid-based approach to calibration, one first needs to project market quotes on the regular and pre-specified grid used for training the neural network. After calibration, one must price quoted strikes and maturities, thus requiring a second run of interpolation/extrapolation. A natural alternative to performing this step is using the 'true' pricing function under the optimal parameter specification. While this makes perfect sense, it could require slow pricing methods. The pointwise approach does not suffer from this drawback.

Regarding rough volatility models, a more technical aspect relates to the specification of the forward variance curve. The standard approach in all mentioned works is to represent the forward curve as a piecewise constant function with (at least) eight levels. Then, one has to calibrate their values jointly with the other model parameters. The approach significantly differs from the earlier proposal in El Euch *et al.* (2019), where the forward variance curve is a state variable, estimated from the variance swaps, and exogenous to the calibration process. One first recovers variance swaps from an infinite log-strip of out-of-the-money options, following Gatheral (2011), and then builds the forward variance curve by differentiation. The curve is piecewise constant, but its identification precedes the calibration step. In this paper, the estimation of the forward curve is part of the calibration procedure. However, we propose a very simple parametrization, which reduces the dimensionality of the calibration problem and guarantees excellent fits to the volatility surface, as we will show extensively.

Our contribution is three-fold. We re-interpret the pointwise approach in a (quasi-)random grid setting, combining the original approaches in Bayer and Stemper (2018) and Horvath *et al.* (2021). The result is a small neural network that calibrates the entire market volatility surface in nearly one second and does not require any interpolation/extrapolation. We provide a neural network-based pricer to be used by traders to evaluate options with every strike-maturity pair within the domain of training of the network itself. Last but not least, we propose a simple parametrization of the forward variance curve, which is also supported by empirical evidence from the market.

Throughout the paper, we will often advocate the concept of a grid that is either fixed, adaptive, or random, depending on the context. We clarify the meaning of these names from the beginning. The fixed grid is the same as the one introduced

by Horvath *et al.* (2021) regarding expirations and strikes. The adaptive grid is a slight modification of the former. We include shorter expirations and make the range of the strikes a function of time to maturity. The reasons for an adaptive grid are apparent after Rømer (2022), but our choice differs from Rømer's. We do not increase the number of maturities in the grid and define our rule for selecting strikes. Finally, a random grid is adaptive, but grid times and strikes are drawn non-uniformly from some contract parameter space (rather than being specified a priori).

Learning from random smiles is also possible and very convenient. We leverage this possibility in the paper. We sample a bunch of maturities for each parameter set, and for each maturity, we price a random vector of strikes. As we will clearly show, this improves the out-of-sample performance of the network (in contrast to the pure pointwise approach) for two reasons. Computationally, generating strikes for a single smile can be performed very efficiently. Second, in the training phase, feeding the network with smiles corresponding to the same parameter set improves learning the conditional density implied by the model.

Grid approaches require two rounds of interpolation/extrapolation—as we noticed. We will always deal with the first one (i.e. projection of the market volatility surface to the grid) via the tools that Vola Dynamics LLC† . kindly made available to us. Such tools are also fundamental for validation of the parametric form of the forward variance curve that we come up with.

We organize the paper as follows. Section 2 introduces the calibration problem and recalls the (rough) models we will deal with in our numerical experiments. Section 3 reviews standard neural network approaches in the literature, underlines their pros and cons and introduces our pointwise approach. Sections 4 and 5 report numerical experiments under the rHeston and rBergomi models and describe the new parametrization of the forward variance curve. We support the importance of an adapted grid for adequately estimating the roughness parameter in the rHeston model and test our methodology against a few volatility surfaces from the market. Section 6 gives possible directions for future research and concludes.

2. The calibration problem

Loosely speaking, the mathematical description of the calibration problem is an optimization as below:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\ d(\sigma_{BS}^{\mathcal{M}}(\boldsymbol{\theta}, K, T), \sigma_{BS}^{mkt}(K, T)), \tag{1}$$

where d is a distance between two sets of points (e.g. the root mean squared error (RMSE)) and $\mathcal{M} = \mathcal{M}(\boldsymbol{\theta})_{\boldsymbol{\theta}\in\Theta}$ is a model with parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p, p \in \mathbb{N}$. The choice to minimize against the market implied volatilities $\sigma_{BS}^{mkt}(K, T)$ is disputable, but typically justified with increased sensitivity of short-dated deep out-of-the-money (OTM) options to model parameters (as opposed to minimizing against option prices). This region is crucial for hedging purposes. One should try to achieve an excellent description of it.

Whenever one knows the CF of the asset log-price under model $\mathcal{M}$, FFT pricing is the obvious choice to perform calibration. This is the standard case with classical stochastic volatility models and first tested against rough models in El Euch *et al.* (2019). However, the problem with the rHeston model is that the CF requires numerical procedures that are too costly to allow for fast calibration, as we are used to seeing with classical Heston, for instance. The pricing function's neural network approximations come naturally in such a setting, and this is even more so when Monte Carlo is the only way, like in the rBergomi model that we will also investigate in this paper.

2.1. *Rough volatility models*

Rough volatility models gathered huge interest during the last few years, especially in response to improved fitting to the volatility surface compared to standard stochastic volatility models. They also naturally provide a theoretical justification of the short-time explosion of the observed ATM skew after Fukasawa (2017) in terms of the roughness parameter.

Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{Q})$ be a filtered probaility space as generated by independent Brownian motions B_t and $B_t^{\perp}$ under the risk neutral measure $\mathbb{Q}$. The evolution of the spot price (the SPX here) undergoes the following dynamics:

$$\mathrm{d}S_t = S_t\sqrt{V_t}(\rho\,\mathrm{d}B_t + \sqrt{1-\rho^2}\,\mathrm{d}B_t^{\perp})$$

upon assumption of zero interest rates and dividends. Here V_t is the variance process and ρ stands for the correlation with the spot price S_t. Different specifications for the variance process give rise to different (rough) models.

2.1.1. rHeston. The rHeston model of El Euch and Rosenbaum (2019) arises with a specification of the variance process as below

$$V_t = \xi_0(t) + \frac{\nu}{\Gamma(H+\frac{1}{2})}\int_0^t \frac{\sqrt{V_s}}{(t-s)^{\frac{1}{2}-H}}\mathrm{d}B_s,$$

where $\xi_0(t) = \mathbb{E}[V_t|\mathcal{F}_0]$ is the time-zero forward variance curve and $\nu > 0$ the volatility of volatility parameter. The parameter H, ranging between zero and a half, represents the roughness parameter of variance paths. As already noticed, the CF of the asset log-price is known up to the numerical solution of a fractional Riccati equation, which can be achieved via the Adams scheme of Diethelm *et al.* (2004), the hybrid scheme of Callegaro *et al.* (2021) or the rational approximation of Gatheral and Radoicic (2019). The latter approach is the one we will chose to generate samples. Alternatively, Abi Jaber (2019) and Abi Jaber and El Euch (2019) discuss Markovian approximations of the variance dynamics. Whatever the route one follows, Fourier techniques are aptly available for numerical integration and we exploit the FFT implementation presented in Cherubini *et al.* (2010) and Baschetti *et al.* (2022), known as the SINC approach.

† Interested readers can contact Vola Dynamics LCC inquiring about an agreement for academic purposes or buy the software by visiting the following webpage: https://voladynamics.com.

2.1.2. rBergomi. The rBergomi model of Bayer *et al.* (2016) corresponds to the following specification of the variance process

$$V_t = \xi_0(t) \exp\left(\eta Y_t^\alpha - \frac{\eta^2}{2} t^{2\alpha-1}\right),$$
$$Y_t^\alpha = \sqrt{2\alpha - 1} \int_0^t (t-s)^{\alpha-1} \mathrm{d}B_s,$$

where $\alpha = 2H - 1$ and $\eta > 0$. Bennedsen *et al.* (2017) and McCrickerd and Pakkanen (2018) present efficient schemes for the simulation of the rBergomi dynamics.

3. Neural networks

Model calibration to real market data has been a target for machine learning algorithms since the pioneering work of Hernandez (2017). This work prompted a flourishing literature that reached a standard with the two-step approach of Horvath *et al.* (2021). For ease of exposition, in this section, we review the two-stage approach, return to the direct approach and finally investigate methods that avoid interpolation/extrapolation.

3.1. *The two-stage approach*

Horvath *et al.* (2021) separate the calibration procedure into two steps:

1. Generate samples according to a model $\mathcal{M}$ and train a neural network to learn semi-parametrically the pricing map $F^{\mathcal{M}}(\boldsymbol{\theta}; w)_{ij} : \boldsymbol{\theta} \mapsto \sigma_{ij} \doteq \sigma(K_i, T_j)$, from the model parameters $\boldsymbol{\theta}$ to implied volatilities σ_{ij} on a fixed strike—time to maturity grid $\{K_i, T_j\}_{i,j=1}^{n,m}$ in terms of the weights w. This amounts to finding neural network weights $w^\star$ that solve

$$w^\star = \underset{w}{\operatorname{argmin}} \sum_{u=1}^{N} \sum_{i=1}^{n} \sum_{j=1}^{m} (F^{\mathcal{M}}(\boldsymbol{\theta}_u, w)_{ij} - \sigma_{BS}^{\mathcal{M}}(\boldsymbol{\theta}_u)_{ij})^2$$

where N is the number of samples in the training set and $\sigma_{BS}^{\mathcal{M}}(\boldsymbol{\theta}_u)_{ij}$ denotes the grid of implied volatilities for a parameter realization $\boldsymbol{\theta}_u$.
2. Solve

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{m} (F^{\mathcal{M}}(\boldsymbol{\theta}; w^\star)_{ij} - \sigma_{BS}^{mkt}(K_i, T_j))^2,$$

where $\sigma_{BS}^{mkt}(K_i, T_j)$ denotes the grid of market implied volatilities for the same strike-time to maturity values.

They set N =68 000 and choose a relatively small grid with $n = 11$ and $m = 8$. As for the network architecture, they use four hidden layers of 30 nodes each. Figure 1 provides a graphical representation of the network.

The grid specification is fundamental in this kind of approach. In Horvath *et al.* (2021), the eleven strike prices are evenly spaced and range from 0.5 to 1.5, while the time to maturity spans the values $\{0.1, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.0\}$ years. Strike prices are independent of the maturity and span a constant region of the asset log-price's probability density function (PDF). We will show that starting the grid at $T = 0.1$ years results in highly biased estimates of the roughness parameter H in the rHeston model. Intuitively, rough models pay off the most in the shortest maturity regimes and allow for an improved fit that standard one-factor models of volatility can never achieve. We will return to this point with numerical experiments in subsection 4.1.

Given the well-known square-root scaling of price volatility, the grid specification in Horvath *et al.* (2021) tends to

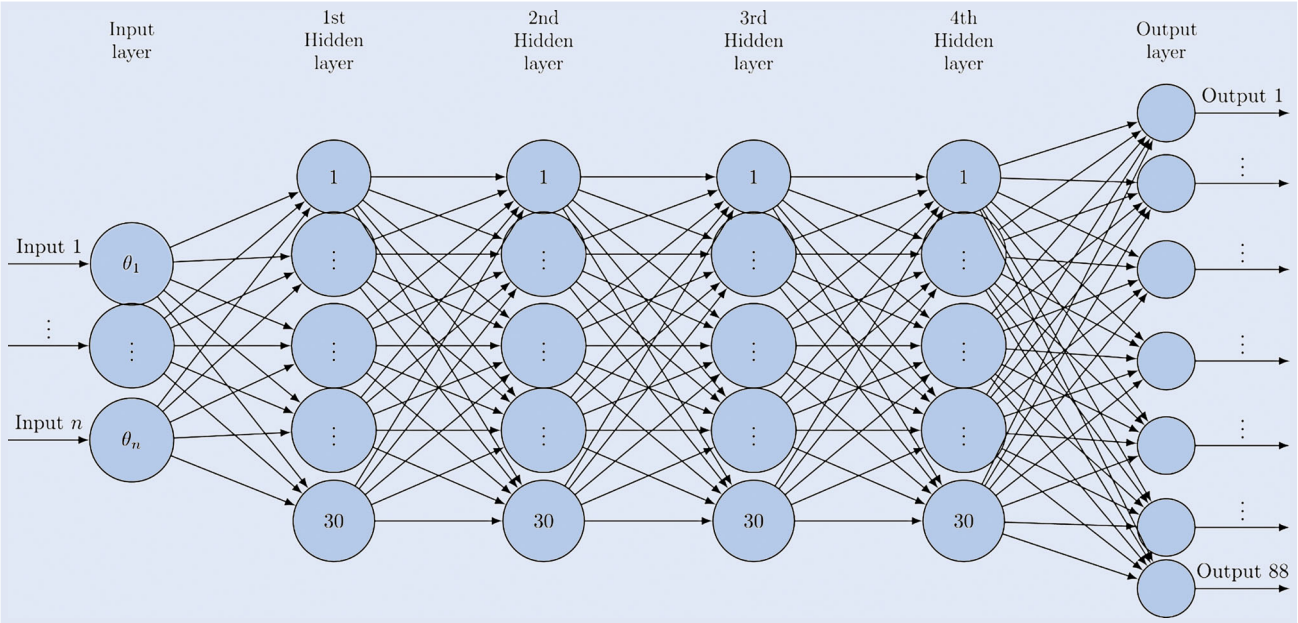


Figure 1. Neural network architecture in Horvath *et al.* (2021).

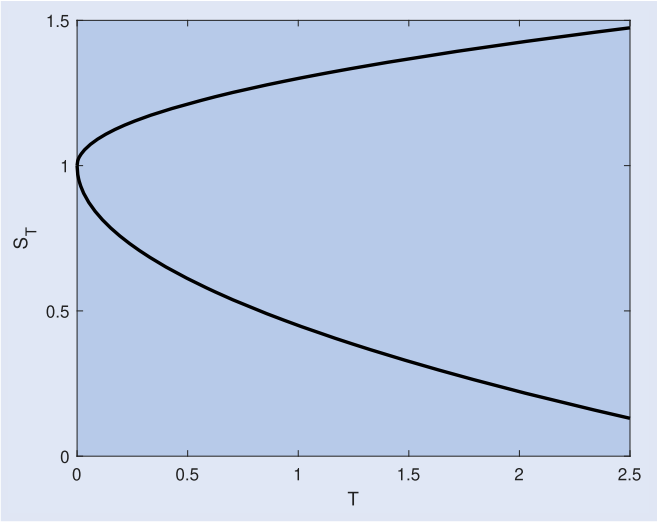


Figure 2. Span of the strike price through time.

over-sample the strike region at short times and under-sample it for long horizons. Similar considerations and practical reasons relating to Monte Carlo errors in the generation of samples prompted Rømer (2022) to an adaptive grid. Making the strike price a function of time to maturity is in fact the only way to guarantee one gathers sufficient statistics in the wings of the smile. In a similar spirit, we also try an adaptive grid. We choose times to maturity $\{0.01, 0.025, 0.1, 0.3, 0.6, 1.0, 1.5, 2.0\}$ years, and, for any T_j, we take (almost) equidistant strikes in the range $[S_0(1 - l\sqrt{T_j}), S_0(1 + u\sqrt{T_j})]$, with $l = 0.55$ and $u = 0.30$. Figure 2 provides a graphical representation of the region covered by the strike price. Please notice that the square root rule only applies until the time to maturity is not too long, at which point strikes would be prescribed negative. Anyway, times are now much more informative and strikes more aligned with market quotes for any given maturity, especially very short ones.

3.2. The direct approach

Hernandez (2017) tackles the calibration problem entirely differently. With some abuse of terminology, we may refer to his approach, compared to the one in Horvath *et al.* (2021), as an inverse one. He exploits machine learning to learn the map from implied volatilities to model parameters. To achieve the goal, he uses a feed-forward neural network with four layers and 64 nodes each. The sample size is larger than Horvath *et al.* (2021) with almost twice as many grid surfaces in the training set. Of course the two approaches share the same training set, with the obvious difference that model parameters and implied volatilities switch their roles. Consistently, the dimension of the input layer is $n \times m$ while the output dimension is p.

3.3. The pointwise approach

Both approaches in Hernandez (2017) and Horvath *et al.* (2021) (and later modifications, e.g. Rømer 2022) require two rounds of volatility data processing. First, one needs to project the input from the market using interpolation/extrapolation routines on the same grid used for training. After calibration, one has to lift the output to the same strike-maturity pairs as the market. The natural choice to perform this step is by calling the true pricing function (by Fourier transform or Monte Carlo simulation) given the calibrated parameters. The last step may only partially preserve the quality of the fit, especially if Monte Carlo is involved, but finer grids would help, as demonstrated in Rømer (2022). Still, the need to run the true pricing function after calibration seems highly inefficient. One could resort to a computationally faster approach based on spline interpolators, but this would be more worrisome when preventing arbitrages and guaranteeing proper extrapolation for short maturities.

The literature supporting the grid-based approach claims that applying interpolation / extrapolation techniques to the implied volatility surface is a well-understood and easy-to-implement procedure. Nonetheless, some words of caution are necessary here: The extrapolation of volatilities at a very short time consistently with the roughness of the market may be subtle. If the smallest expiration in the grid is shorter than the first maturity in the market, a flat extrapolation would bias the roughness parameter high, and the longer the first maturity, the higher the bias. While a flat extrapolation is overly simplistic, the design of a polynomial extrapolator consistent with the implied volatility asymptotic behavior dictated by the model is generally not trivial. Dall'Acqua *et al.* (2023) and Bourgey *et al.* (2023) present a volatility extrapolation procedure consistent with a rough model's dynamic properties. As it will soon become apparent, our calibration procedure proposes an alternative and entirely numerical solution.

3.3.1. Pointwise training. Bayer and Stemper (2018) and Liu *et al.* (2019) introduce the pointwise approach. This is also two-stage, but the idea is that the network learns single points in volatility rather than a grid and calibrates directly to the market. By design, the trained network can generate implied volatility without the need to interpolate over a specified grid in strike and time-to-maturity. Unfortunately, however, the very large shape of the network makes calibration much slower than in the grid-based approach. For training the network, the authors sample each combination of model parameters $\{\boldsymbol{\theta}_u\}_{u=1,\dots,N}$ concurrently with one strike-maturity pair (K_u, T_u) from some contract parameter space. Therefore they need to run as many Monte Carlo simulations or Fourier inversions as the number of samples. For adequate training of a large network—the number of nodes per layer ranges from a few hundred to about four thousands—the number of samples must be as high as 10^6, thus making the entire training phase quite time consuming.

3.3.2. Training with random grids. Our approach is of pointwise type too, since we are learning the underlying map from model parameters and contract specifications to (single points in) implied volatility. Figure 3 indeed clarifies that we inherit the same network structure as in a pointwise method:

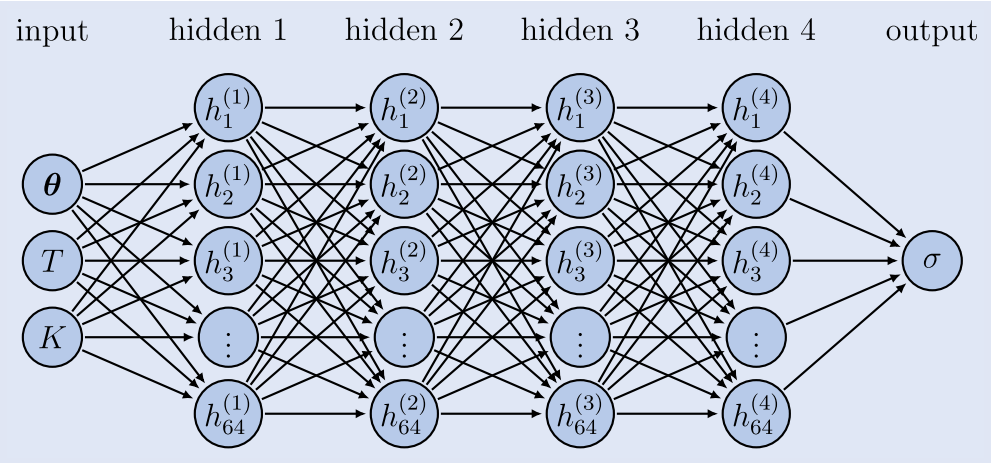


Figure 3. Neural network architecture as we use it in the numerical section. Boldface notation underlines that $\boldsymbol{\theta}$ stands for the entire parameter vector: the input layer is $p + 2$-dimensional, where p is the number of model parameters.

The NN takes model parameters $\boldsymbol{\theta}$, strike K and time to maturity T in input, and it outputs the implied volatility σ of the associated option.

Nonetheless, we introduce an important difference with respect to Bayer and Stemper (2018) and Liu *et al.* (2019). The key is that the training set that we dispense to the network results from the following procedure: We produce grid surfaces in the generation phase but pass them to the network in the form of quadruplets $(\boldsymbol{\theta}_l, K_i, T_j, \sigma_{ij})_{l,i,j=1}^{q,m,n}$ (we have $q \times m \times n$ of them). Generation is clearly reminiscent of Horvath *et al.* (2021) but strikes and maturities are no longer restricted to a pre-specified fixed (or adapted, Rømer 2022) grid.

Indeed, there is no particular reason why times should be confined to a few buckets of maturity and strikes predetermined given the expiration. Both of them can be taken random, and we explain how to do it in the rest of this section.

We sample parameters sets $\boldsymbol{\theta}_l, l = 1, \ldots, q$ uniformly at random from Θ and associate each particular set with $n = 11$ expiries and $m = 13$ strikes. In particular, maturities $T_1^l, \ldots, T_n^l$ are drawn from the following set

$$\begin{aligned}[T_{\min}, T_{\max}] = {} & [0.003, 0.030) \cup [0.030, 0.090) \\ & \cup [0.090, 0.150) \cup [0.150, 0.300) \\ & \cup [0.300, 0.500) \cup [0.500, 0.750) \\ & \cup [0.750, 1.000) \cup [1.000, 1.250) \\ & \cup [1.250, 1.500) \cup [1.500, 2.000) \\ & \cup [2.000, 2.500], \end{aligned} \tag{2}$$

T_j^l coming from the jth subinterval ($j = 1, \ldots, n$). This last fact ensures proper coverage of the typical market maturities in the training set. Finally, strikes $K_1(T), \ldots, K_m(T)$ are sampled in range whose width is a function

$$\begin{aligned}[K_{\min}(T), K_{\max}(T)] = [S_0(1 - l\sqrt{T}), S_0(1 + u\sqrt{T})], \\ l = 0.55 \quad u = 0.30. \end{aligned} \tag{3}$$

of time to maturity T.

We provide a graphical description of the dataset construction procedure below:

$$\begin{cases} \boldsymbol{\theta}_1 \begin{cases} T_1^1 \longrightarrow K_1(T_1^1) \ \ldots \ K_m(T_1^1) \\ \vdots \\ T_n^1 \longrightarrow K_1(T_n^1) \ \ldots \ K_m(T_n^1) \end{cases} \\ \vdots \\ \boldsymbol{\theta}_q \begin{cases} T_1^q \longrightarrow K_1(T_1^q) \ \ldots \ K_m(T_1^q) \\ \vdots \\ T_n^q \longrightarrow K_1(T_n^q) \ \ldots \ K_m(T_n^q) \end{cases} \end{cases}.$$

Crucially, strikes are not uniform in $[K_{\min}(T), K_{\max}(T)]$ but our sampling tries to replicate similar granularity as the market. More specifically, we take

- four strikes in the left tail $[K_{\min}(T), S_0(1 - 0.20\sqrt{T})]$
- seven strikes in the central region $[S_0(1 - 0.20\sqrt{T}), S_0(1 + 0.20\sqrt{T})]$
- two strikes in the rigth tail $[S_0(1 + 0.20\sqrt{T}), K_{\max}(T)]$

Our grids are therefore adaptive in nature but promoting them to be random allows for rearrangement in a form that is suitable for the pointwise approach, which fact is enough to calibrate on market points directly and getting rid of any interpolation/extrapolation. Notably, then, we will also see in the numerical section that such a calibration is typically very fast.

We will talk about a random-grid pointwise approach to denote our hybrid solution and distinguish it from the original proposals in Bayer and Stemper (2018) and Liu *et al.* (2019) - which could now be casted as truly pointwise (thus meaning that both generation of samples and submission to the network are indeed pointwise, i.e. each parameter set is associated with one strike-maturity pair only).

The benefits from our method are multiple:

- production of the training set is very fast compared to an approach that is truly pointwise. Say one wants to produce $M = N_T \times N_K$ points

overall when describing a surface (N_T maturities, N_K strikes per maturity) and assume—for the moment—that the CF of the asset log-price is known (e.g. the rHeston model): individual options can be priced with N calls to the CF. Populating the whole surface only requires $N \times N_T$ calls to the CF with our random grid approach, as opposed to $N_T \times N_K \times N$ if one is truly pointwise. A similar reasoning applies when pricing by Monte Carlo methods. Random grids require one simulation on a time grid that contains all of the desired maturities, but a truly pointwise approach would ask for $N_T \times N_K$ simulations.

- proper training only needs a small number of neurons, for a network which is comparable in size with the grid-based approach from Horvath *et al.* (2021). Numerical experiments in this paper are based on a neural network with 4 hidden layers, 64 nodes each. The reason why such a thin network structure performs properly is investigated in the following subsection.

Options falling outside of the contract parameters space that we used in the generation phase cannot be priced by the network. However, our square root rule for the selection of strikes is such that what is left over is typically quite illiquid and would be excluded in any case. Also, the same choice for the range of the strikes is also needed in a grid-based approach.

Finally, we notice that calibration with a pointwise method is very similar to what one does with the FFT (and would do with Monte Carlo). A fundamental difference remains. Every network approximation of the pricing function is much faster to evaluate than the function itself, and the calibration process faster as a consequence.

3.3.3. Training with random smiles. Another viable choice for pointwise training comes with the generation of random smiles. This strategy is indeed halfway between the pure pointwise approach and training with random grids.

We sample parameter sets $\boldsymbol{\theta}_l$ $l = 1, \dots, q$ from space Θ; we supplement each of these sets with random time to maturity T and strikes $K_1(T), \dots, K_m(T)$; we price the corresponding options and invert for implied volatility. Expirations are such that we guarantee uniform coverage of time sub-intervals in equation (2). The usual square root bounds (3) define the range of the strikes. Finally, we dispense the generated smiles to the network in a pointwise manner.

A systematic comparison of the out-of-sample performance between the pure pointwise approach and training with random smiles is illustrated in figures 4 and 5 for the rHeston model.

We start with increasing the number of points in the training set and look at the evolution of the NN approximation error. Figure 4 shows that there is a level in the size of the training set where a network which is purely pointwise stops learning: We see no decreasing trend in the blue candles after 2^{19} training samples have been included. Conversely, training with random smiles (in orange) exhibits monotonic decrease of the average error throughout the whole region we are considering. The two approaches behave pretty similarly for small $\log_2(N_{train})$, but the gap gets larger and larger as we proceed.

We therefore conclude that filling the parameter space in a purely pointwise way is not effective for proper training.

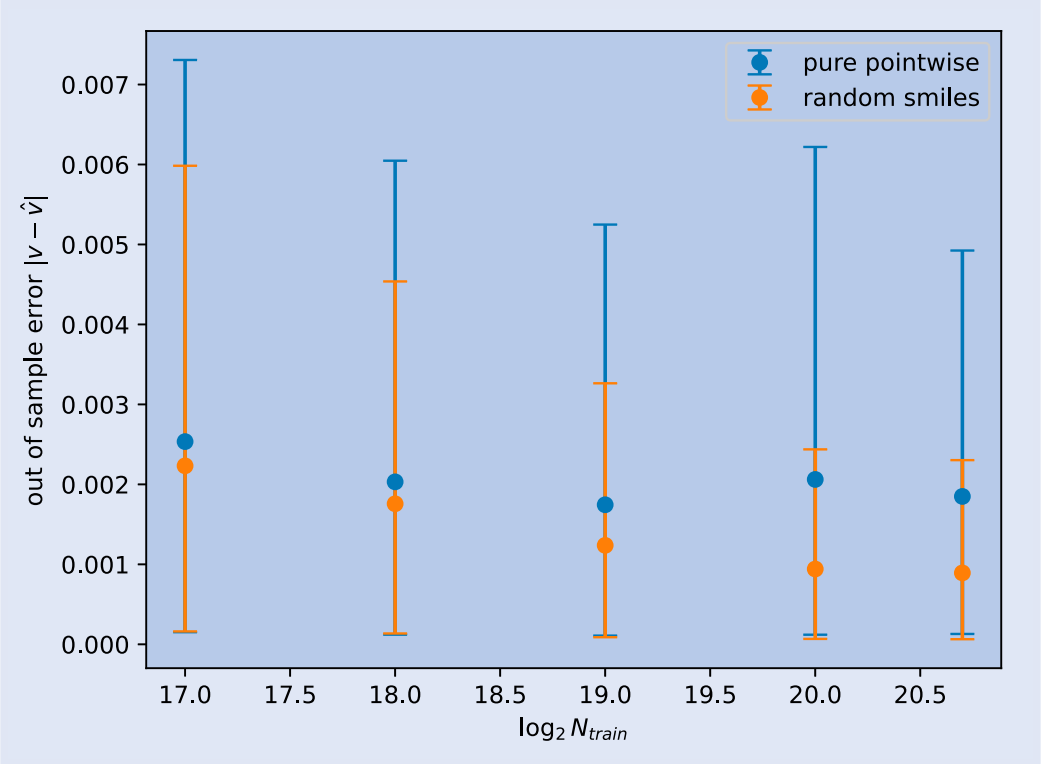


Figure 4. Evolution of the out-of-sample error as a function of the number of data points in the training set. Pure pointwise approach in blue, training with random smiles in orange. Candles cover the range between the 5th and 95th quantiles. Points denote the mean absolute error.

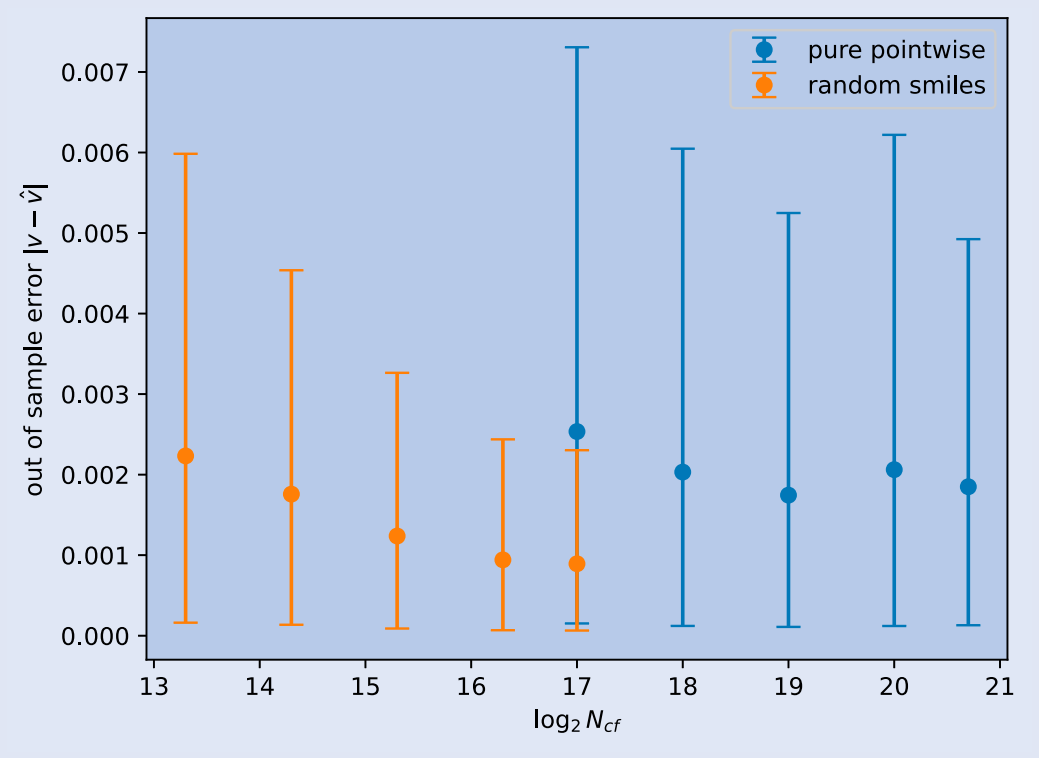


Figure 5. Evolution of the out-of-sample error as a function of the computational burden, measured in terms of calls of the CF, N_{cf}. Pure pointwise approach in blue, training with random smiles in orange. Candles cover the range between the 5th and 95th quantiles. Points denote the mean absolute error.

Most importantly, the network needs to be exposed to multiple strikes so as to learn information about the smile, i.e. about the conditional density implied by the model. Proper coverage of typical market maturities also plays a role.

Figure 4 also suggests that proper training can be achieved without increasing the size of the network. Layers and nodes are fixed throughout the experiment, but we are able to achieve excellent out-of-sample performance when sampling random smiles. We can work with a much thinner network than Bayer and Stemper (2018) and Liu *et al.* (2019) because of the information we include in the training data. Generating multiple entire volatility surfaces on random grids but training as if they were pointwise provides the network with rich information about the shape of the smiles and their evolution across different maturities.

The size of the training set is a crucial variable to account for, but what is more important is the computational cost of the entire generation phase (i.e. building samples for the training set). Specifically for those models where the CF is available in semi-closed form, one can perform a fair comparison between the pointwise and the random smile approach in terms of the computational burden, by counting the number of times the algorithm computes the CF. Plotting the NN approximation error as a function of the number of calls of the CF in figure 5 makes the superior performance of the random smile approach even more apparent. For $N_{\text{cf}} = 2^{17}$, the average error from the pure pointwise approach is larger than the 95th quantile of the corresponding orange candle.

Generating random smiles rather than individual points guarantees a large gain in computational time (roughly m times faster, where m is the number of strikes in one smile). The CF only depends on the model parameters and time to maturity, and can be re-used for all strikes in the random smile approach, without the need to recompute it for each point as in the pointwise approach.

A similar reasoning can be extended to purely Monte Carlo models, such as the rBergomi, where the whole cost comes from simulation through maturity T but is obviously almost insensitive to the number of options priced in a single smile. However, a proper investigation of the relative performances of the pointwise and random smile approach in a Monte Carlo setting may depend on the implementation details and model at hand. We do not consider it in the present work and leave the analysis for future research.

3.3.4. Practitioner's corner: NN architecture and training. The object we will be working with in the numerical section is a simple feed-forward neural network with 4 hidden layers, 64 nodes each.† Model parameters $\boldsymbol{\theta}$ and contract specifications (T, K) enter as an input, and the associated implied volatility comes in the output. A graphical description of the network has been presented in figure 3.

The Elu activation function

$$a_{ELU} = \begin{cases} x & \text{if} x \geq 0 \\ \mathrm{e}^x - 1 & \text{if} x < 0 \end{cases}$$

is solely responsible for the non-linearity in the network. The output layer comes with the identity function as an activation:

† We did not try any optimization on the network structure and therefore simpler architectures could provide satisfactory results as well.

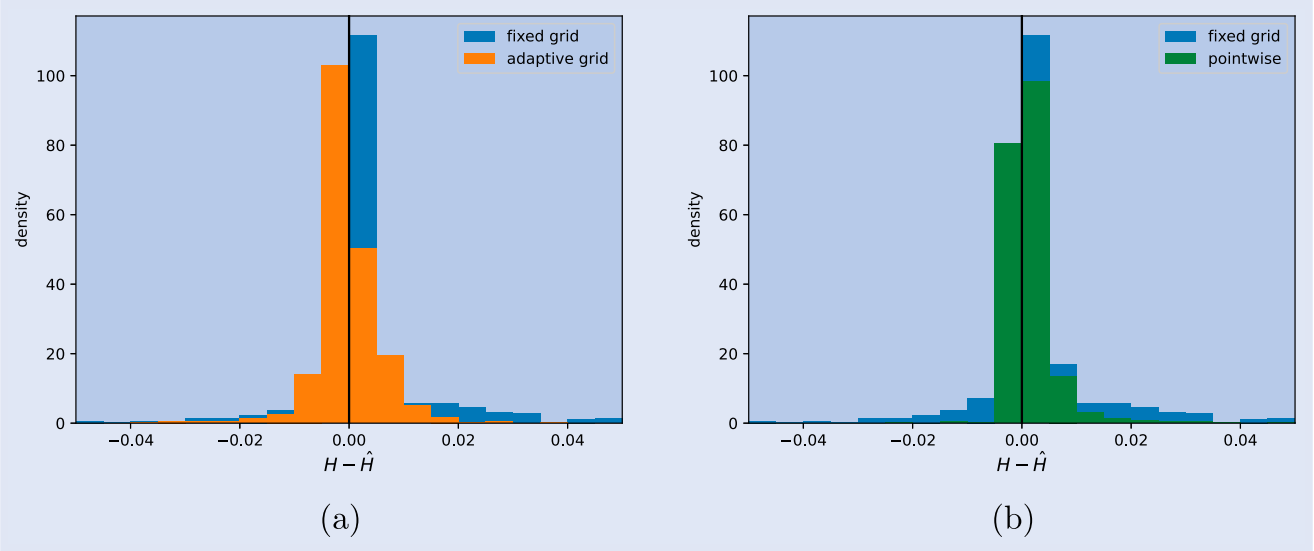


Figure 6. Distribution of absolute errors e_H for fixed-, adaptive- and random-grid neural network estimates of the roughness parameter H in the rHeston model over a sample of 1000 surfaces with flat variance curve.

we find that there is no need to enforce positivity of the implied volatility in any way.

We use the RMSE as a loss and take advantage of Adam optimizer during training. We allow for 500 epochs at most, and impose a patience parameter of 50 epochs for early stopping. The validation loss is monitored. No dropout or more advanced techniques seem to be needed to prevent overfitting. Mini batches are used for performance. Other hyperparameters are unchanged with respect to the default in Keras.

Once the network is trained,† we save neural network weights to disk and use the optimal configuration for calibration. The neural approximation of the true pricing function and its gradient are hard-coded in Numpy. This allows for extra speed relative to the prediction method in Keras, as suggested by Horvath *et al.* (2021).

A working example for the usage of the network is available on the following GitHub page: https://github.com/fabioBaschetti/random-grid-NN-calib.

4. Numerical experiments

This section presents a large set of tests for the random-grid pointwise approach. We show how it naturally emerges as a solution to various problems from a grid method (be it a fixed grid or adapted), and equip it with a parametric forward variance curve before we draw our conclusions.

The standard with rough volatility models is that the forward variance curve is piecewise constant. Moreover, when it comes to neural network methods for pricing and calibration it is common practice to associate grid times with the discontinuities of the curve. We stick to this convention. As for the pointwise approach, we make the forward variance curve entirely equivalent to the one we use for the adaptive grid (i.e. same buckets).

4.1. *Wily estimate of the roughness parameter*

Horvath *et al.* (2021) apply fixed grid methods for calibration of the rBergomi model. We also try rHeston and spend a few words of caution for such an application in the present section.

We already noticed that extension to an adaptive grid is crucial for inclusion of very short expirations in the training set. In particular, for the rHeston model, this also results in better estimates of the Hurst exponent that are key to capture the explosive behavior of the ATM skew, as we now demonstrate.

For this, we sample a thousand parameter sets, compute implied volatilities over both a fixed grid and an adaptive grid, and calibrate using the appropriate network. The underlying forward variance curve is piecewise constant but jump times synchronized with grid maturities and therefore different. Benchmark volatilies that we calibrate on are generated with a flat forward variance curve. Having a third construction of the curve that both networks could replicate is indeed necessary not to favor any of the two and ensure that model parameters are not biased by forward variance effects—at least in principle.

All considerations in the present subsection find graphical support in the following plots. We refer the reader to appendix 1 for the numbers behind them.

While both networks seem to do a good job, a few points are worth noticing after figure 6 and the corresponding table A1. The range of the absolute errors $e_H = H - \hat{H}$ is much larger for the fixed grid than it is for the adapted grid and the tails of the error distribution fatter. Also, very large negative errors are observed with the fixed grid (greatly larger than the largest positive error), thus resulting in huge overestimation of the roughness parameter. Short maturities carry

† Proper training clearly depends on the quality of the samples one has produced during the generation phase. We do not enter the technicalities behind Monte Carlo estimation of rBergomi option prices (for which we refer to Bennedsen *et al.* 2017, McCrickerd and Pakkanen 2018), but stress that those parameter combinations that produced prices for which Black-Scholes inversion fails need to be discarded.

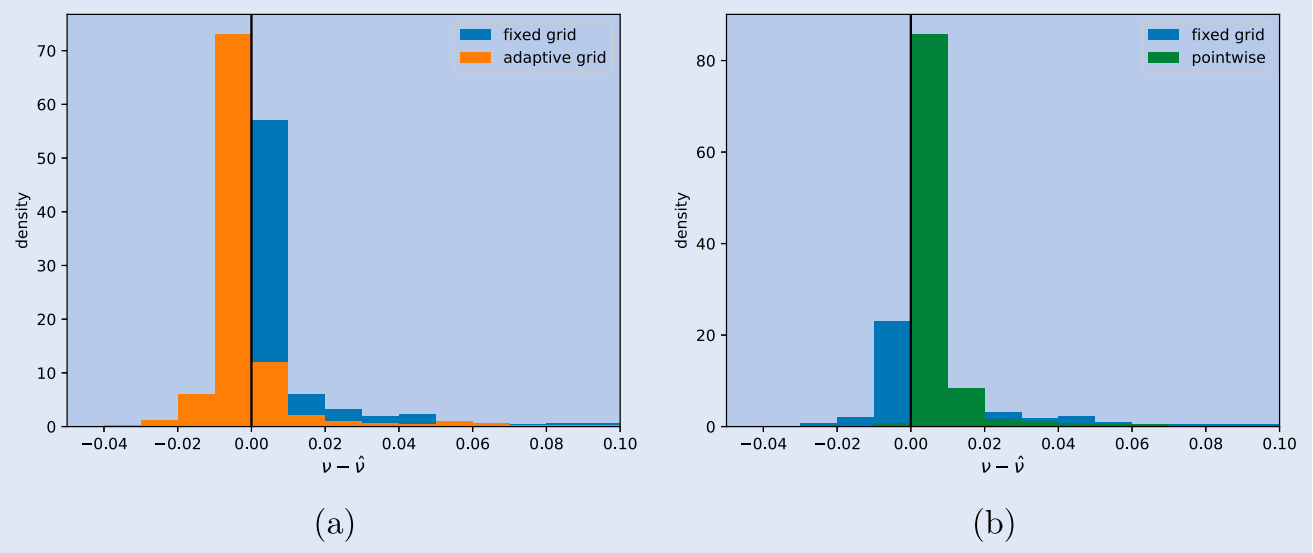


Figure 7. Distribution of absolute errors e_ν for fixed-grid, adaptive-grid and pointwise (adaptive-grid) neural network estimates of the VOV ν in the rHeston model over a sample of 1000 surfaces with flat variance curve.

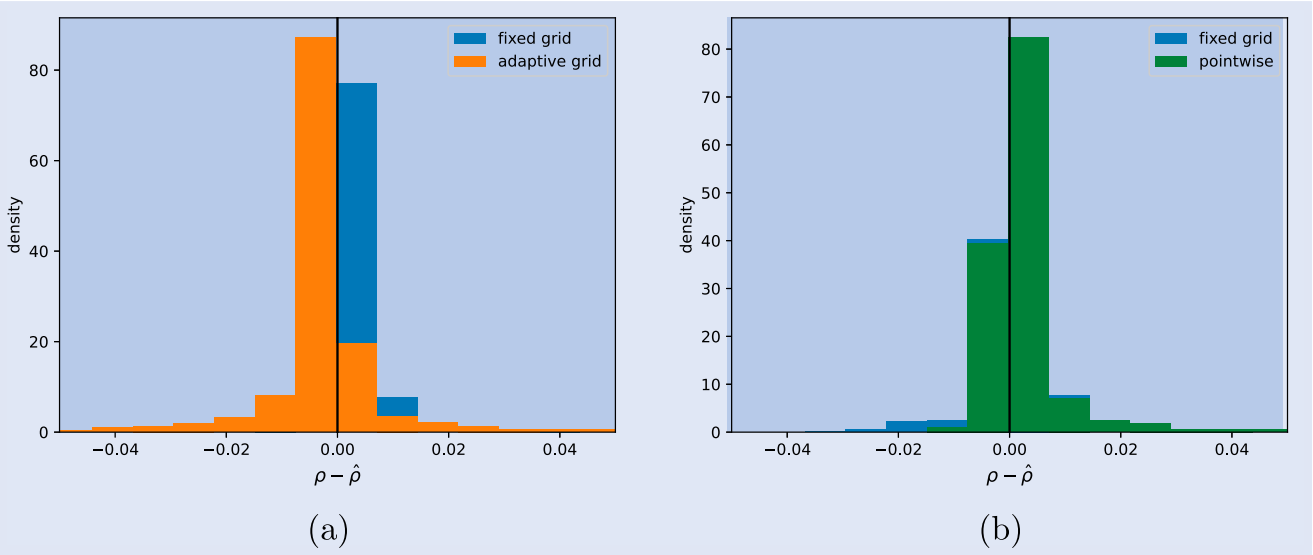


Figure 8. Distribution of absolute errors e_ρ for fixed-grid, adaptive-grid and pointwise (adaptive-grid) neural network estimates of the correlation ρ in the rHeston model over a sample of 1000 surfaces with flat variance curve.

important information about the skew and kurtosis; ignoring them biases our estimates of the roughness parameter as a consequence.

The lowest variability in the estimation of H is actually observed with the random-grid pointwise approach. Standard deviation of the error is slightly better than the adaptive grid, and a factor three smaller than the fixed grid (see table A1).

The other parameters—volatility of volatility (VOV for short) and correlation—require separate treatment. The average error in the estimation of ν and ρ gets closer to zero when moving to the adapted grid but no reduction in the standard deviation is observed. Actually, visual inspection of figure 7(a) would suggest a general overestimation (underestimation) of the VOV with the adapted (fixed) grid. It is indeed true that the estimates from the adapted grid are higher than the true value of the VOV about 80% of the times (vs 26% for the fixed grid) but the average error in table A2 is very close to zero thus suggesting that such an overestimation is mild. In addition to this, the pointwise approach has (almost) always a positive bias and we should remember that it is now using the same set of points for calibration as the adapted grid. It is therefore unlikely that this erratic behavior of the overestimation ratio when moving from one network to another is explained by the particular set of expirations and strikes that we have selected. Most importantly, then, the average error is always small and the pointwise approach is the one solution than makes the standard deviation the smallest—as it is now evident in figure 7(b) and confirmed in table A2.

As for the correlation parameter ρ, the adaptive grid exhibits more variability than the fixed grid (figure 8(a)) but the pointwise approach fixes this (figure 8(b)). Table A3 reveals that the average error with the pointwise approach is very similar to the fixed grid but the range much smaller and the standard deviation halved.

Hence, the pointwise approach emerges as the best compromise from this comparison, but its true potentialities will be

further explored in the subsequent sections. For the moment, we only conclude that making the range of the strikes depend on the maturity of the option is not only more practical under simulation, but more financially sound and particularly important to get an accurate estimate of the roughness of the volatility. We believe this is an important piece of information as an entire stream of literature is especially interested in the subject: from possible estimation methods to the description of the empirical behavior of the ATM skew.

4.2. Pricing off the grid

Whenever a grid-based method is used for calibration, separate pricing of all market points follows. Points off the grid may be achieved either via interpolation techniques or with the true pricing function. Either ways, the estimation issues we have outlined above impact on the quality of the fit.

Because the pointwise approach calibrates to all available data (at least those in the domain where the network has been trained) we expect it to perform better than the combination of a grid-based method plus extension to the market points. Then, it is natural to take the pointwise approach as a benchmark. Evidence in the previous section suggests that the general tendency from the fixed grid is an underestimation of the roughness parameter being accompanied with lower volatility of volatility and more negative spot-vol correlation. Market data confirms this tendency. March 16, 2021 is an example, with the following calibration parameters in table 1 for the rHeston model:

Table 1. Estimated rHeston parameters as of March 16, 2021 using both types of grid and the pointwise approach.

Model parameters	Fixed grid	Adaptive grid	Pointwise
H	0.0100	0.0933	0.0808
ν	0.3563	0.5065	0.5355
ρ	−0.8085	−0.6575	−0.6678

Extension to all market points with the true pricing function is biased as a consequence of the calibrated parameters from the grid being biased. Specifically, the roughness parameter H being lower than it should results in a steeper smile at very short times. Not only, problems may also arise with the level of the smile. The optimal level of the forward variance curve for the first grid maturity is not necessarily optimal for shorter market times thus shifting the smile vertically. Both effects are visible in the upper left corner of figure 9 (green vs cyan lines).

Poor quality of the fit also extends to longer maturities. It is important to notice, though, that the real matter with such later times is not extension itself but the quality of the calibration to

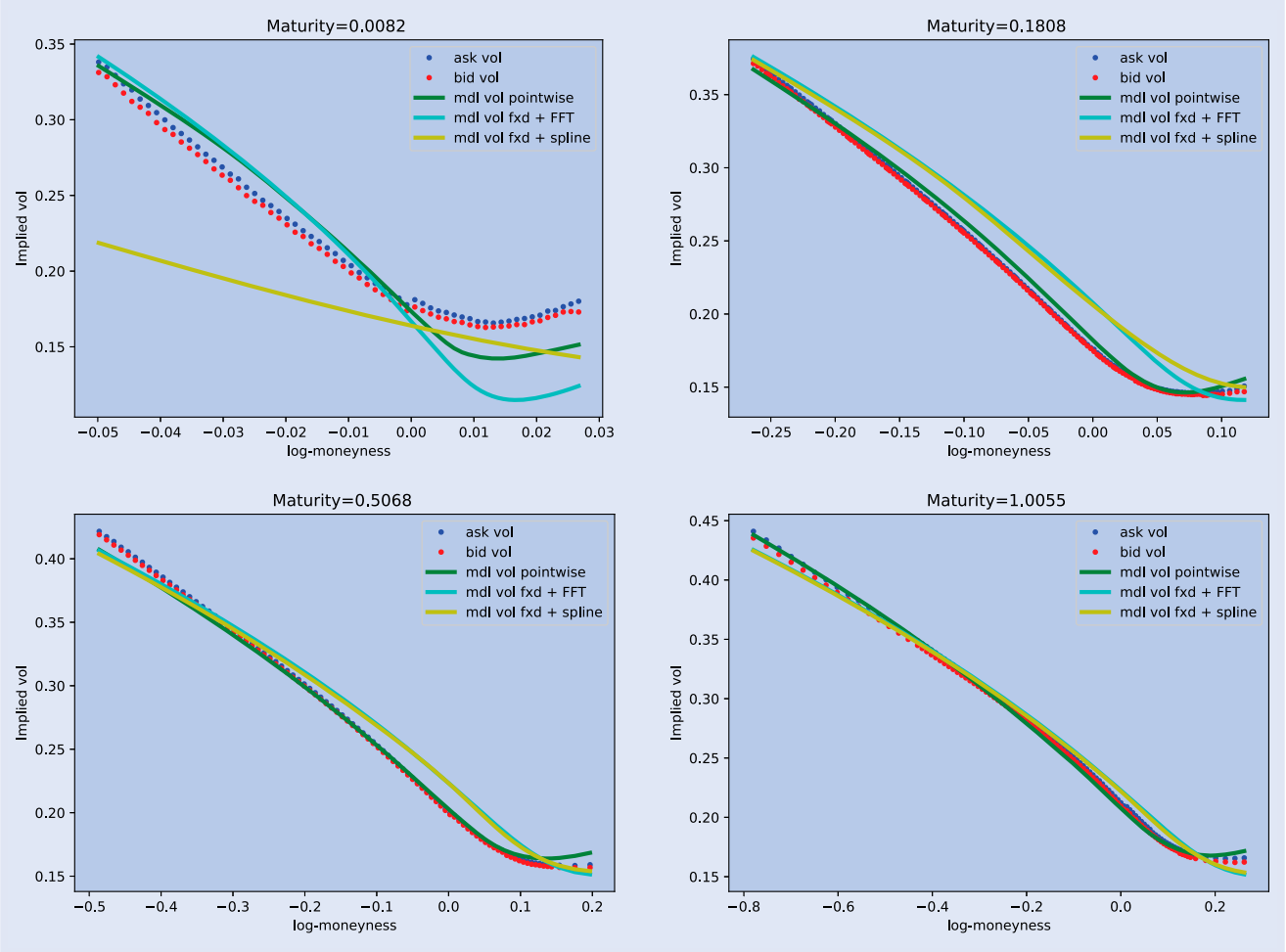


Figure 9. Calibration to the market vol. surface as of March 16, 2021. Green is pointwise (random grid), cyan is fixed grid (fxd) plus interpolation/extrapolation by the true pricing function (FFT for rHeston), and yellow is fixed grid plus splines.

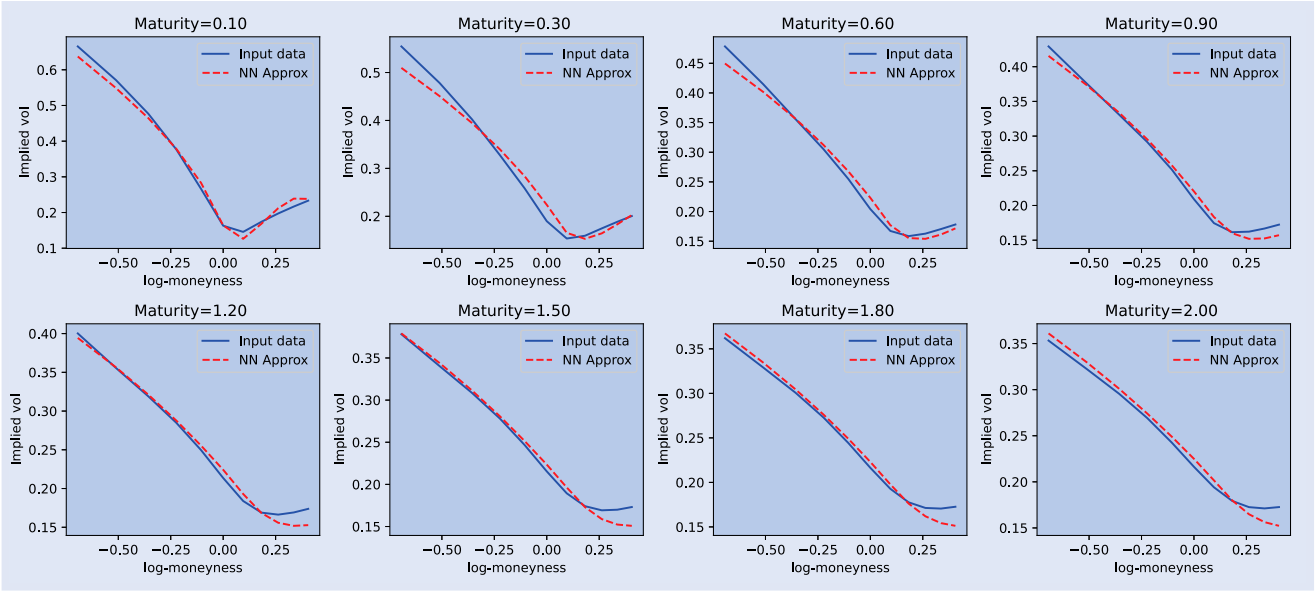


Figure 10. Neural network calibration of the market projection on a fixed grid as of March 16, 2021.

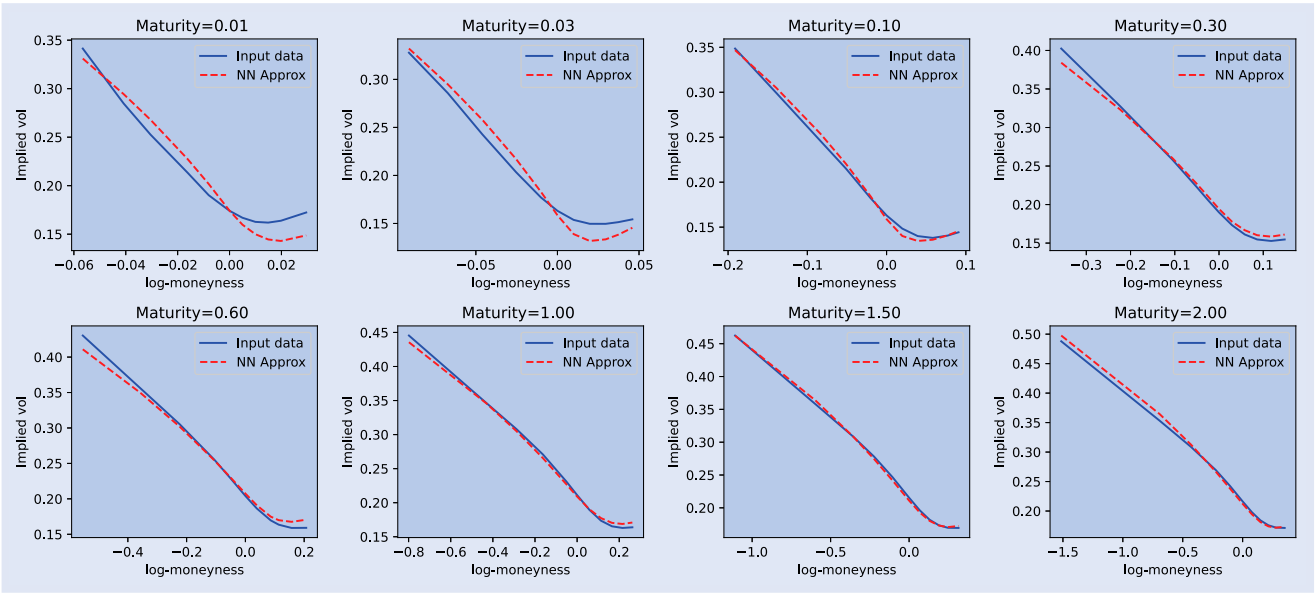


Figure 11. Neural network calibration of the market projection on an adapted grid as of March 16, 2021.

the grid. If the fit to the grid is not good enough, extension to the whole market cannot certainly do better. Figure 10 in fact reveals substantial difference in the shape of the smile when comparing grid volatilities with the fit from the network, especially around the ATM. Needless to say, problems on the grid propagate to the whole market irrespectively of the method one uses for extension. The yellow curves in figure 9 that we obtain via interpolation with spline techniques are also very far from the market.

Extrapolation via spline techniques needs a special mention. Standard practice in the literature is to extrapolate the smiles flat before the first maturity, but doing so from 0.1 years may result in short-dated smiles to loose all of their curvature. This effect is clear in the top left corner of figure 9 (yellow curve).

All in all, then, a fixed grid must be recognized to be possibly inappropriate for calibration to the market for at least two reasons. Firstly, it does not allow for the inclusion of very short expiries that are crucial for proper estimation of the roughness and extrapolation via the true pricing function. Secondly, minimizing a distance over implied volatilities and making the range of the strikes so large for short maturities as a couple of months or so places much of the weight on the wings of the smile at the cost of the ATM region. The price for this is poor interpolation as we have described above.

The case of an adaptive grid is very much different. Model parameters are now closer to those we obtain with the pointwise approach—if we come back to table 1. Also, the forward variance curve being more versatile at short times† partially fixes possible issues with the level of the first smiles.

The fact that including short-dated options provides us with improved estimates of the roughness parameter is no surprise

† Recall that we have included very short maturities in the adaptive grid and ensured the forward variance curve changes level over the same buckets.

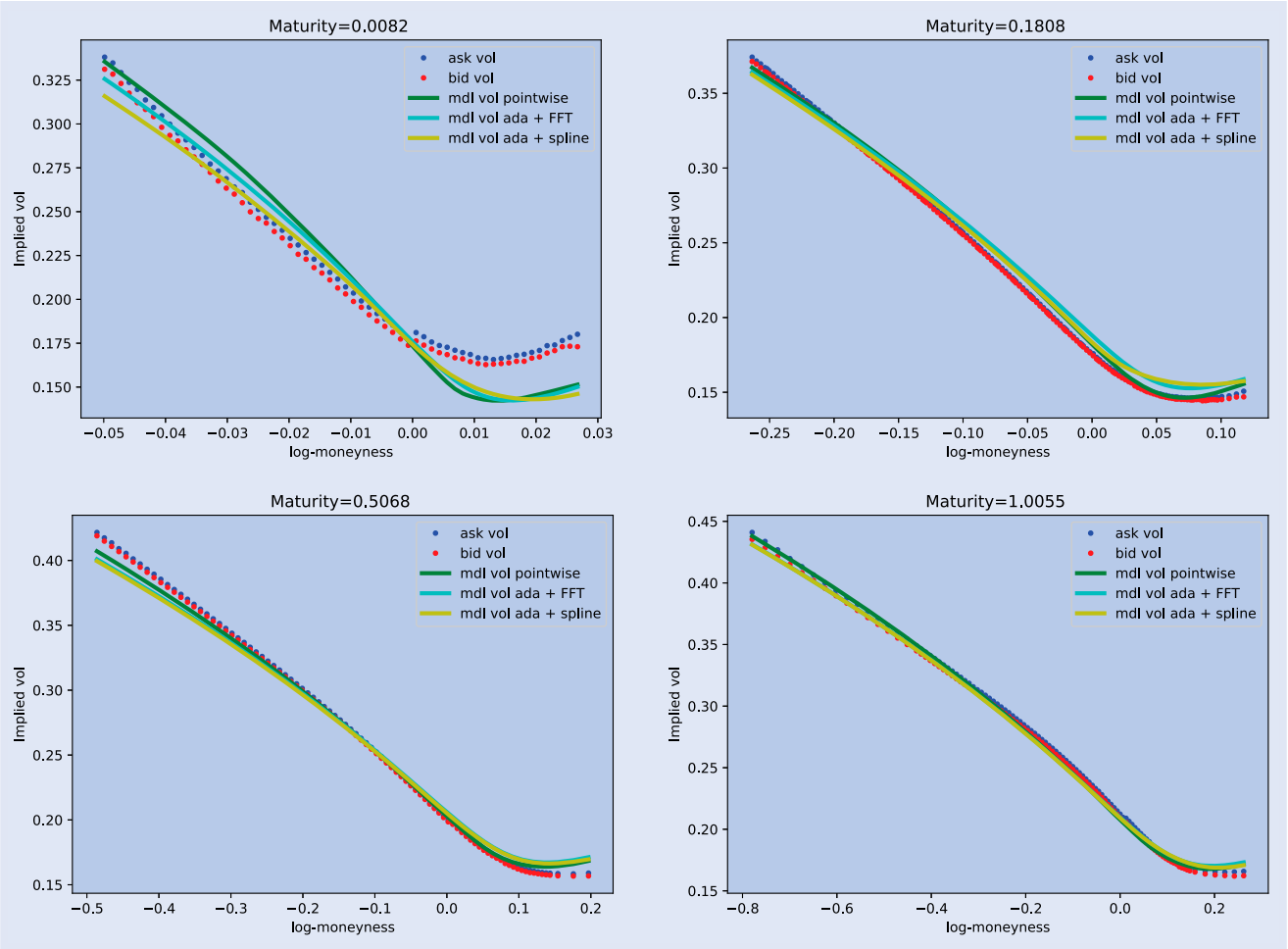


Figure 12. Calibration to the market vol. surface as of March 16, 2021. Green is pointwise (random grid), cyan is adaptive grid (ada) plus interp./extrap. by the true pricing function (FFT for rHeston) and yellow is adapted grid plus splines.

after the discussion in the previous section. VOV and correlation are also more sound and the overall fit to the market is definitely superior with respect to the one we are able to achieve with a fixed grid.

Figure 11 shows those problems around the ATM that we reported with a fixed grid have now disappeared for monthly maturities and longer. We therefore expect interpolation to be grounded on a solid base and do its job when moving to the market volatility surface. This is confirmed in figure 12 smiles are not as good as with the pointwise approach (at least not always) but far better than they used to be with a fixed grid.

Extrapolation now needs to bridge a much shorter interval and it therefore performs better than before. Actually, it happens, in this particular case, that the local performance of a mixture method grid plus interpolation/extrapolation is superior to the pointwise approach: The yellow curve in the top left corner of figure 12 fits the market slightly better than the green one.

This is clearly a random effect which cannot be made systematic and cannot become a global phenomenon. Conversely, theoretical reasons for the pointwise approach to perform better are obviously found in the fact that it sees all of the dataset, by its nature.

In addition to this, different extrapolation methods suffer from different problems. The true pricing function is too slow to be competitive and spline techniques could make the smile too flat at very short times. The adaptive grid alleviates this problem but does not solve it entirely. July 17, 2014 is an example of this as seen in figure 13.

We therefore conclude that the pointwise approach is more grounded and robust than a grid approach and, most importantly, it is self-contained, in the sense that it does not require any additional tools but it deals with both pricing and calibration alone (and in very decent time).

4.3. Forward variance specifications

Option pricing and calibration under stochastic volatility models like rHeston and rBergomi—that we consider here—require precise choices for the treatment of the forward variance curve.

One can build it from variance swaps† and make it a state variable following the example of El Euch *et al.* (2019) or—and this is more standard nowadays—one just fixes c points in time and calibrates this many levels of a step-function

† Variance swaps can be priced in terms of an infinite log-strip of out of the money European options as explained in Gatheral (2011). The idea is that one comes up with a parametrization of the volatility surface and uses it to extend the market to a continuum of option prices to integrate over. Once variance swaps are priced on the market maturities, the forward variance curve can be readily computed by differentiation.

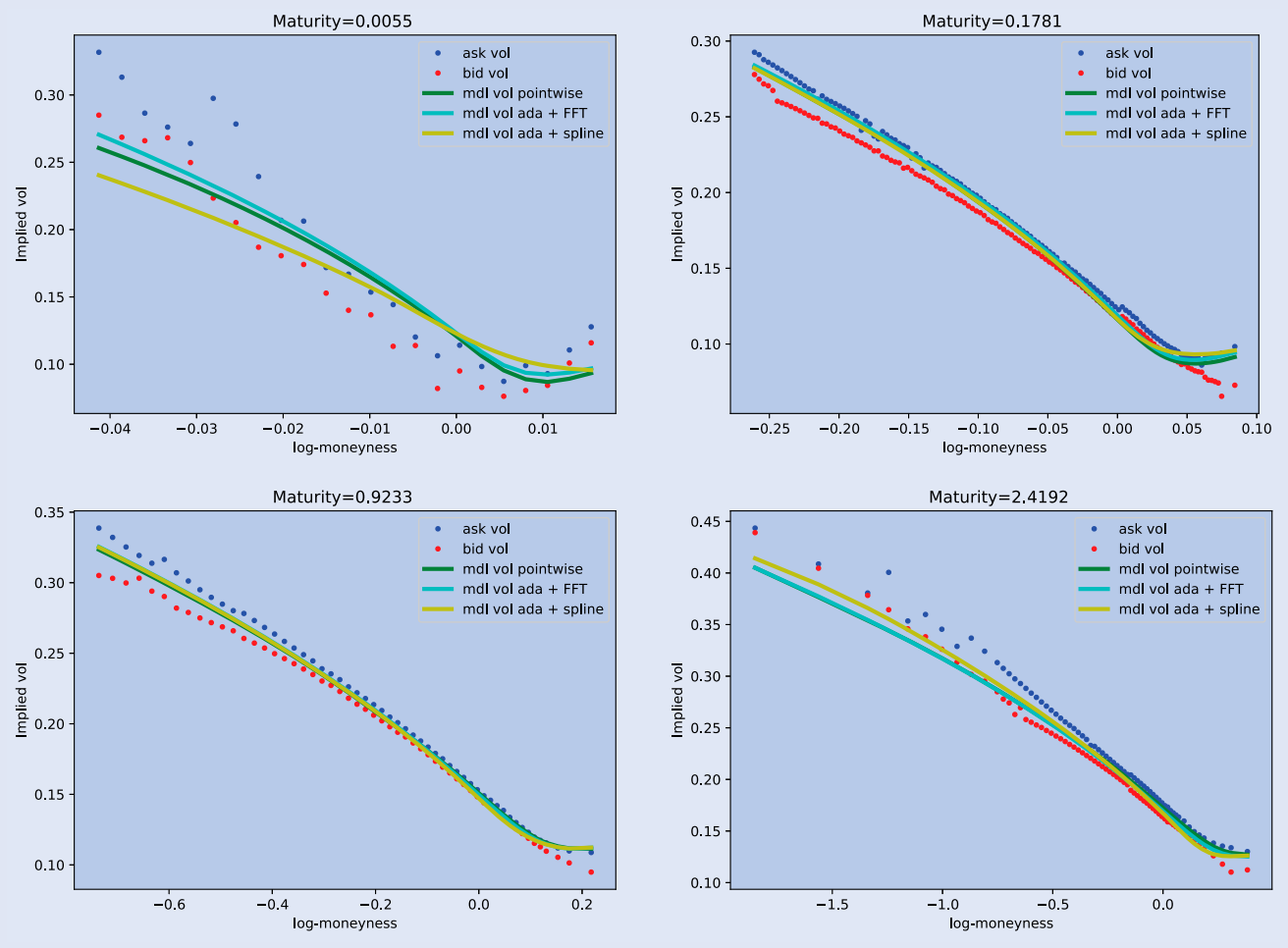


Figure 13. Calibration to the market vol. surface as of July 17, 2014. Green is pointwise (random grid), cyan is adaptive grid (ada) plus interpolation/extrapolation by the true pricing function (FFT for rHeston) and yellow is adapted grid plus splines.

together with model parameters. Such a number c may be significantly reduced with respect to the market maturities with almost no effect on the quality of the fit. Also, making the forward variance curve to jump at specified times opens to neural network pricing of rough volatility models. The dimensionality of the calibration problem is clearly increased but this is largely justified by the boost one obtains from the network—as compared to FFT inversion or Monte Carlo.

In the same spirit of the forward variance curve as a calibration object, we propose a simple parametrization that also comes with very good fits to the market.

The logic behind this is that we calibrate the rHeston model to a number of volatility surfaces for different days in history between 2010 and 2022 and plot the estimated piecewise constant forward variance curve to get a sense of the 'typical' shape.

We report a few such curves as an example in figure 14 and recognize that the vast majority of the surfaces only exhibits a little bit of a term structure at short times—if any. Then, monotonic growth as in figure 14(a) may be handled with a straightforward exponential form but the presence of humps and/or bumps in figure 14(b) requires functional forms that are somehow more involved, for example:

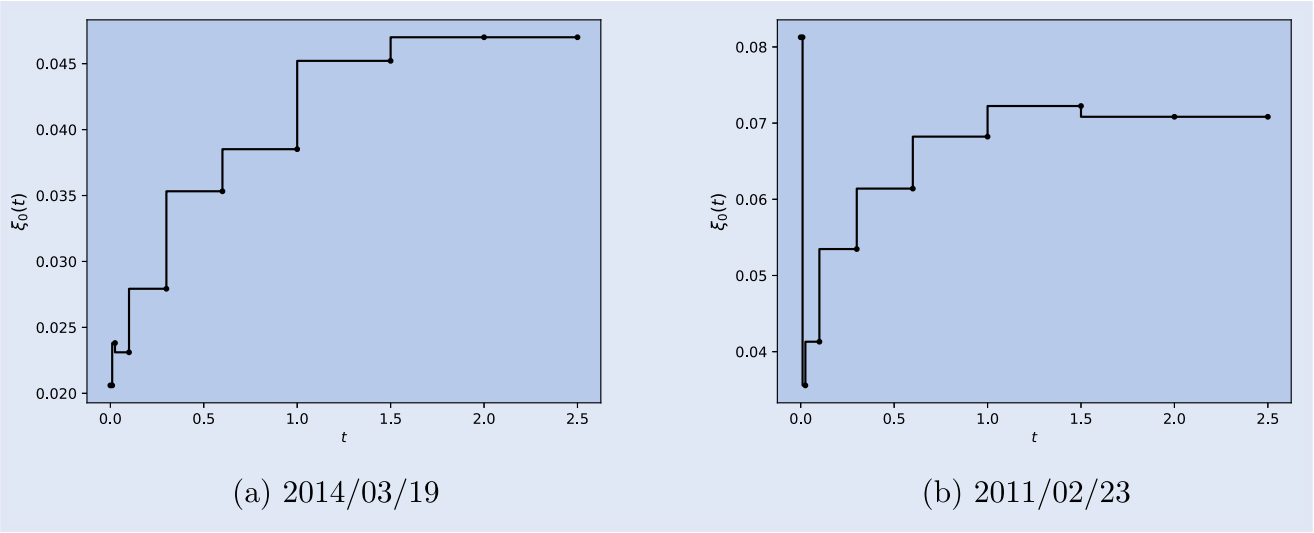
$$\xi_0(t) = \beta_0 + \beta_1 \exp\left(-\frac{t}{\tau_1}\right) + \beta_2 \left(\frac{t}{\tau_2}\right) \exp\left(-\frac{t}{\tau_2}\right) \quad (4)$$

so that

$$\lim_{t\to 0} \xi_0(t) = \beta_0 + \beta_1 > 0$$
$$\lim_{t\to +\infty} \xi_0(t) = \beta_0 > 0.$$

The reader would recognize in our equation (4) a slight generalization of the Nelson and Siegel (1987) model, where the long-term component e^{-x} and the mixed term xe^{-x} are now allowed to evolve at a different rate. It is precisely the interplay of those two terms that is responsible for the humps and/or bumps that we were trying to replicate. For this, we observe that two stationary points are allowed under equation (4) when $\beta_1\beta_2 < 0$ and τ_2 prescribes sufficiently fast increase/decrease at very short times.

We identify the mixed component to be a short-term one because we confine τ_2 to a region such that the contribution from this is exhausted in about one year. In particular, we take forward variance parameters uniformly at random from the hypercube in table 2 for generation and training. We therefore calibrate over this same region during step 2. We do not waste computer time dealing with parameter sets where (at least one of) the β coefficients are approximately zero, because the associated curves in those regions are practically indistinguishable from one another. Yet, the network will most likely



(a) 2014/03/19 (b) 2011/02/23

Figure 14. Estimated piecewise constant forward variance curve as a result of calibration with the pointwise (random grid) approach under the rHeston model. (a) 2014/03/19 and (b) 2011/02/23.

Table 2. Range of the forward variance parameters.

Forward variance parameters	
β_0	$\mathcal{U}([0.025, 0.160])$
$\beta_0 + \beta_1$	$\mathcal{U}([0.005, 0.130])$
β_2	$\mathcal{U}([-.150, 0.250])$
τ_1	$\mathcal{U}([0.001, 1.350])$
τ_2	$\mathcal{U}([0.001, 0.125])$

Table 3. Optimal rHeston parameters under a piecewise constant (pwc) and parametric (par) forward variance curve as of 2014/03/19.

	H	ν	ρ
pwc	0.0454	0.3054	−0.6434
par	0.0496	0.3076	−0.6514

Table 4. Optimal rHeston parameters under a piecewise constant (pwc) and parametric (par) forward variance curve as of 2011/02/23.

	H	ν	ρ
pwc	0.0803	0.3777	−0.7209
par	0.0780	0.3701	−0.7233

be able to bridge this gap and let us calibrate over the whole hypercube.

Because nothing prevents our specification of the forward variance curve to become negative for particular choices of the parameters, we simply discard those combinations while producing the samples and only train the neural network on well-behaved parameters sets. The true market curve being strictly positive, this restriction should be no worry.†

We test the adequacy of this parametrization by checking the optimal fit with the one obtained with a piecewise constant forward variance curve (8 levels and the synchronized with grid times in our adaptive grid). Not only figures 15 and 16 show that the fits from the two approaches are typically very similar but the model parameters always consistent, as one can see from tables 3 and 4.

If this seems to be convincing about the potentiality of equation (4) as a description of the forward variance curve—in the rHeston model, for now—we should maybe spend a few words commenting the practical need for a parametric form to slavishly follow the calibrated piecewise constant curve. The reader may indeed have encountered forward variance curves that exhibit much more of a jumpy behavior at short times compared to what we have plotted in figure 14 and may be worried about the impossibility for our modeling framework to accommodate for this. While reducing the number of levels introduces some regularization where market maturities are denser, we still sometimes observe term structures which are particularly rich during the first month or so. Figure 17 is a clear example for this, but it also demonstrates that all those ups and downs in the forward variance curve are not essential for proper fit to the market. We place ourselves right after the double pick in the piecewise constant curve at time $T = 0.0822$ and observe that the generated smile is incredibly similar to the one we get with a parametric forward variance curve which is much more flat over the relevant period $(0, T]$.

Because a 'flat' initial term structure could have been achieved by a piecewise constant forward variance curve as well, the insensibility of the smile to the shape of the curve is to be read as a sign of possible local minima in the objective function. We indeed recall that the log of the characteristic function in the rHeston model is a convolution between a fractional Riccati equation and the forward variance curve (up to the maturity of the option), which is therefore acting

† We indeed observe the calibrated forward variance curve to be negative at very short times (only) if no such short maturities as a week or so are included in the volatility surface, in which case the surface would be of reduced practical interest (no price for typical hedging options, no clue as to the exploding behavior of the ATM skew and so on). In any case, a non-linearly constrained optimization may be set up to avoid problems of this sort.

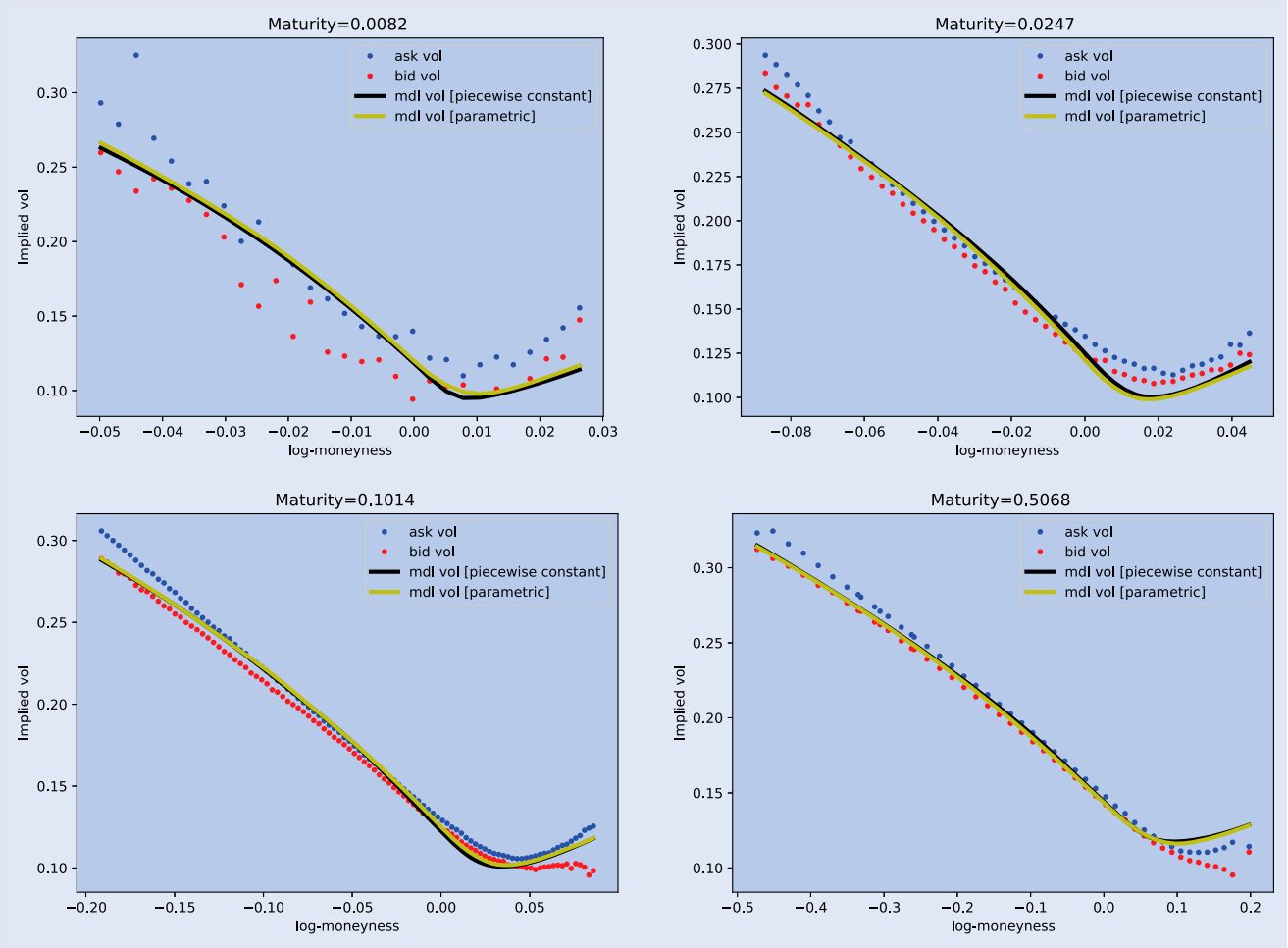


Figure 15. Comparing optimal rHeston fits (selected smiles) under a piecewise constant specification of the forward variance curve (black) vs the parametric model of equation (4) (yellow) as of 2014/03/19.

as a set of weights. Not surprisingly, then, different weights may combine to produce similar option prices whenever more than one forward variance level concur to the price itself. The other face of this phenomenon with our parametric form in equation (4) is that different parameters may produce similar curves. Anyway, consistently with market standards and the view of the forward variance curve as a state variable, we actually only care about the shape of the curve and do not worry about the parameters that have produced it.

Also because the log of the characteristic function is a convolution, the value of the forward variance curve at large times gets less and less important relative to the whole past of the curve. It may consequently happen—in the insufficiency of long maturities from the market—that the calibrated levels at large times seem to be inconsistent with the previous history of the curve (because of a sudden drop/increase, for example). Our parametric form will not be fooled by this inconvenience and still produces very good fits.

There is therefore no need for us to conceive more complicated parametric forms in the attempt to follow the entire zoo of possible piecewise constant forward variance curves.

The reason why we started with the rHeston model is that the forward variance curve seems to play much more an important role here than it does in rBergomi. In fact, the calibrated parametric forward variance curve is typically very different from its piecewise constant counterpart but such a difference is not visible in terms of the fits to the market implied volatilities. We omit any visual representation of this fact—in the interest of space—but recognize that it will certainly result in increased freedom for the modeler to come up with a valuable shape of the curve.

On account of this, we propose the same parametrization in equation (4) for the rBergomi model as well.

Notably, however, we will be able to provide evidence for the validity of our functional form in a way which is completely model-independent in a few pages.

5. Our recipe at work

The need for an interpolation/extrapolation-free neural network-based pricer—together with the evidence from the previous sections—suggests using a random-grid pointwise approach. Making the forward variance curve a parametric function may also be useful when dealing with rough volatility models. We will always have this construction in mind when talking about a pointwise approach in this section.

We therefore devise a last round of tests for the specific framework described above. The idea is that we place ourselves under the rHeston model and check consistency of our

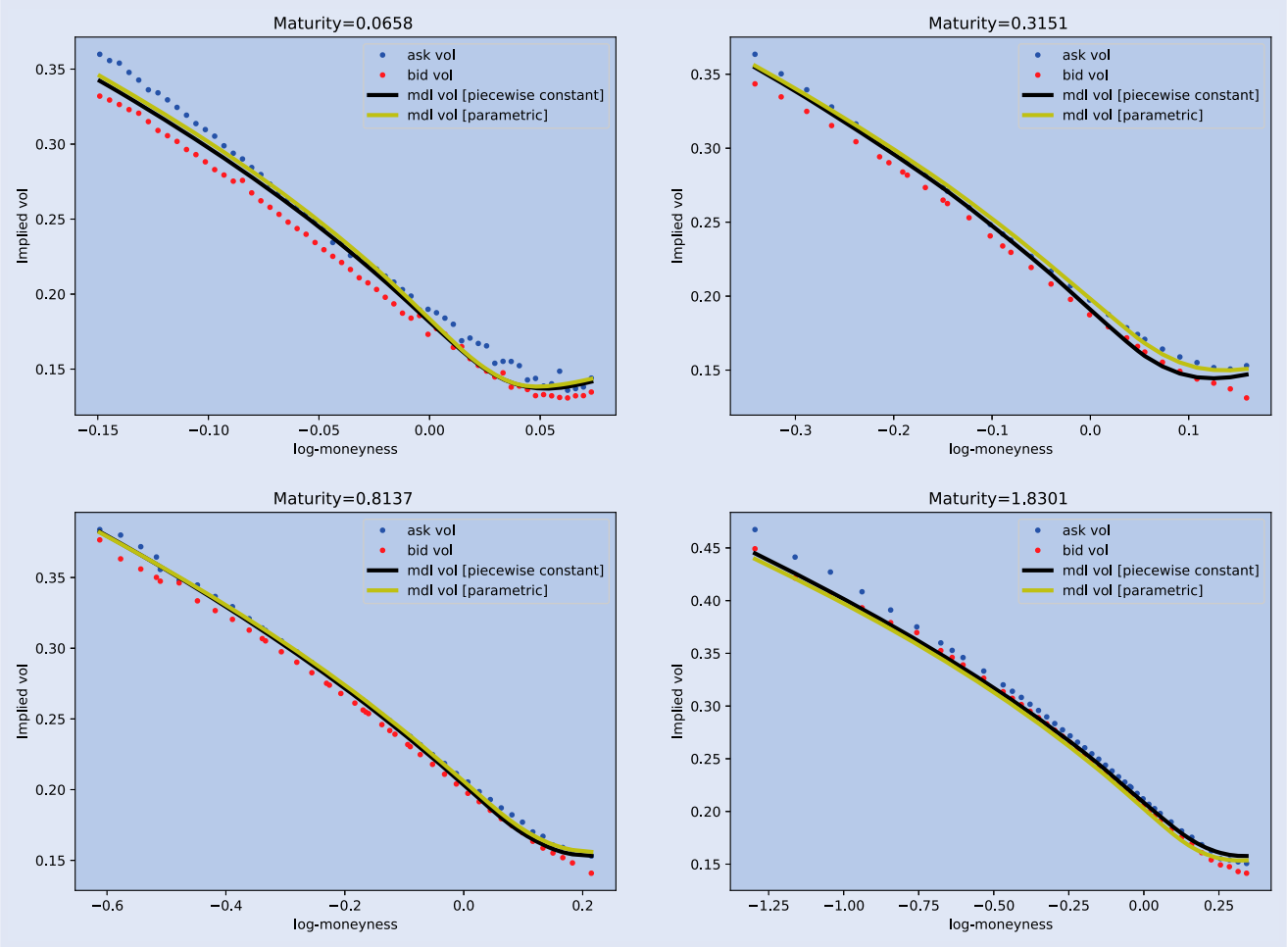


Figure 16. Comparing optimal rHeston fits (selected smiles) under a piecewise constant specification of the forward variance curve (black) vs the parametric model of equation (4) (yellow) as of 2011/02/23.

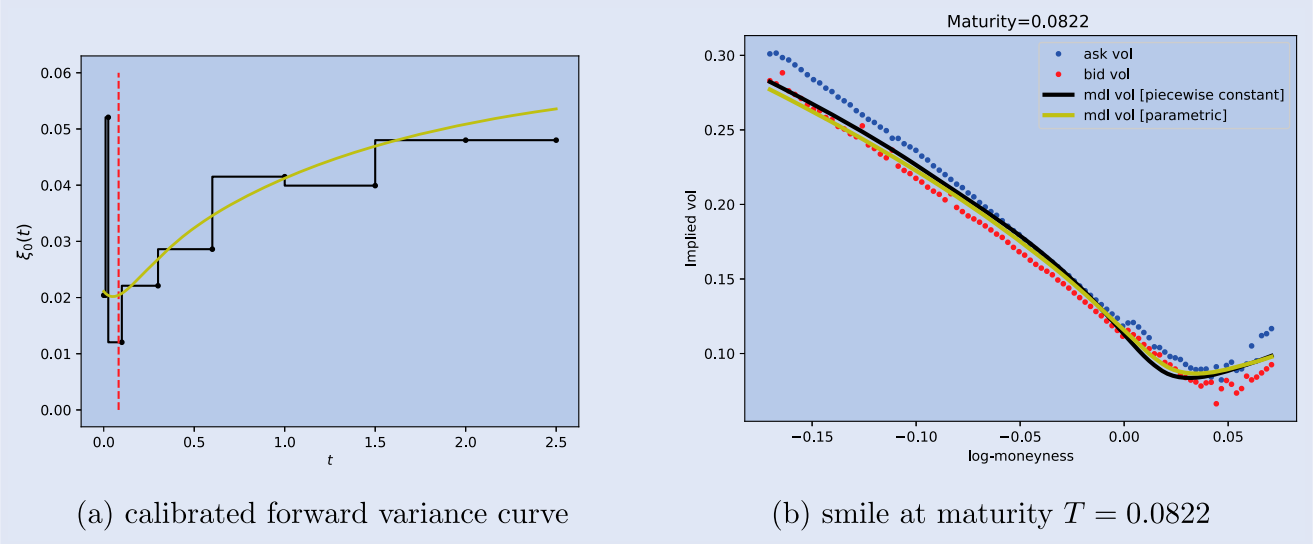


Figure 17. Calibrated piecewise constant (black) and parametric (yellow) forward variance curve as of 2014/07/17 and associated fit to the market at maturity $T = 0.0822$ under the rHeston model. The red dashed line indicates the position of the selected maturity on the forward variance curve. (a) Calibrated forward variance curve and (b) Smile at maturity $T = 0.0822$.

solution with the same procedure as described in El Euch *et al.* (2019). We derive a piecewise constant forward variance curve from quoted implied volatilities, pass it to an FFT pricer and optimize over the model parameters alone. The FFT method is taken as a benchmark, and the performance of our NN pointwise approach tested against it. We also superimpose optimal rBergomi fits for visual comparison, but a systematic description of the models' ability to calibrate to the volatility

Table 5. Domain of definition of the models parameters (rHeston and rBergomi) as seen from the neural network.

rHeston parameters		rBergomi parameters	
H	$\mathcal{U}([0.01, 0.25])$	H	$\mathcal{U}([0.025, 0.50])$
ν	$\mathcal{U}([0.15, 0.65])$	η	$\mathcal{U}([0.50, 4.00])$
ρ	$\mathcal{U}([-.95, -.50])$	ρ	$\mathcal{U}([-.95, -.10])$

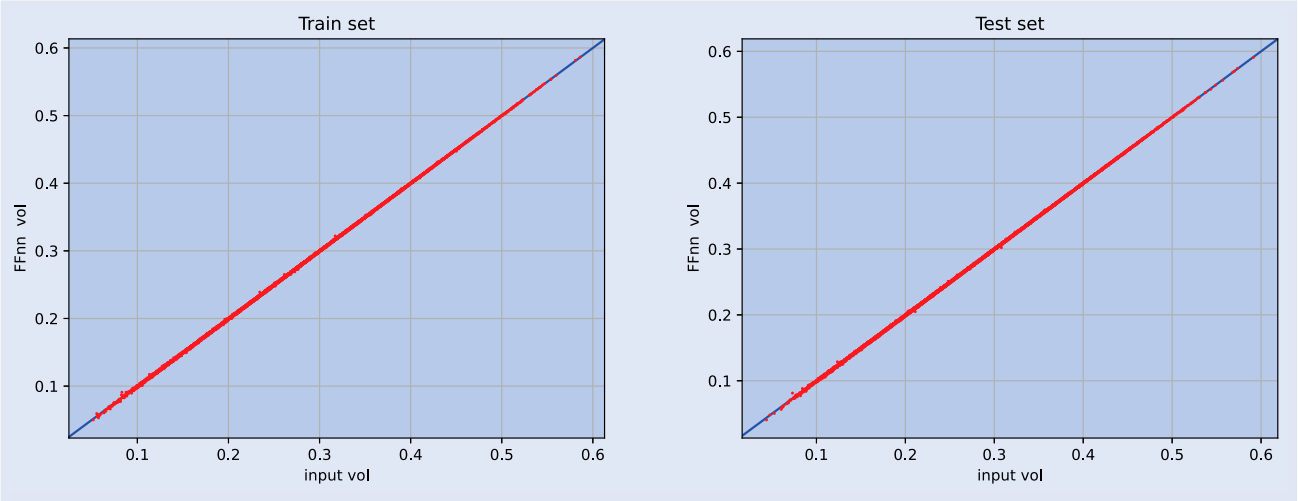


Figure 18. (Model, network) point pairs vs 45-degree line for the rHeston model with parametric forward variance curve. In-sample performance on the left, out-of-sample on the right.

surface under different market scenarios is beyond the scope of this paper.

We restrict ourselves to the following set of model parameters in table 5 for the experiments that follow.

If the quality of the calibration is obviously important, the time associated with it is also a variable that we need to keep an eye on. For this, we believe comparison with the large adapted grid of Rømer (2022) and a version of the pointwise method where the forward variance curve is piecewise constant are relevant for the discussion.

Before we start, however, it is maybe the case to understand how good the network is in replicating the underlying pricing function that we are passing it and generalizing beyond the training set. Figures 18 and 19 show that the network does its job for the models we are considering, with (model,network) point pairs almost perfectly lying on the 45-degree line both in-sample and out-of-sample.

Because the pricing functions associated with the rHeston and rBergomi are free of arbitrage by construction and we have reached such an accurate approximation of them, we are confident that the well-trained NN is not prone to any arbitrage opportunity. We report in appendix 2 an additional robustness check supporting our conclusion and the reliability of the entire procedure.

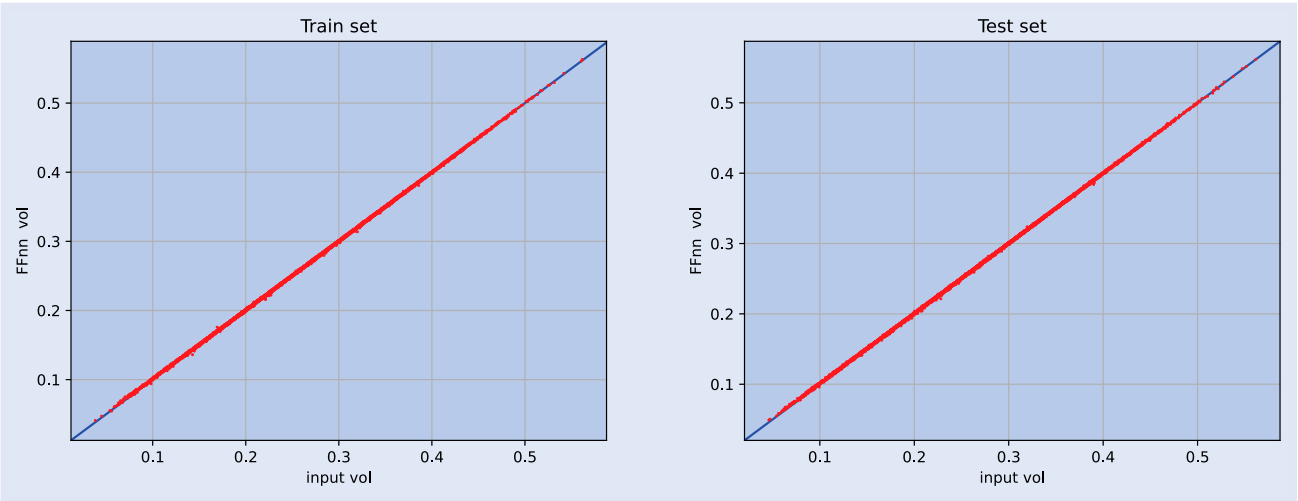


Figure 19. (Model, network) point pairs vs 45-degree line for the rBergomi model with parametric forward variance curve. In-sample performance on the left, out-of-sample on the right.

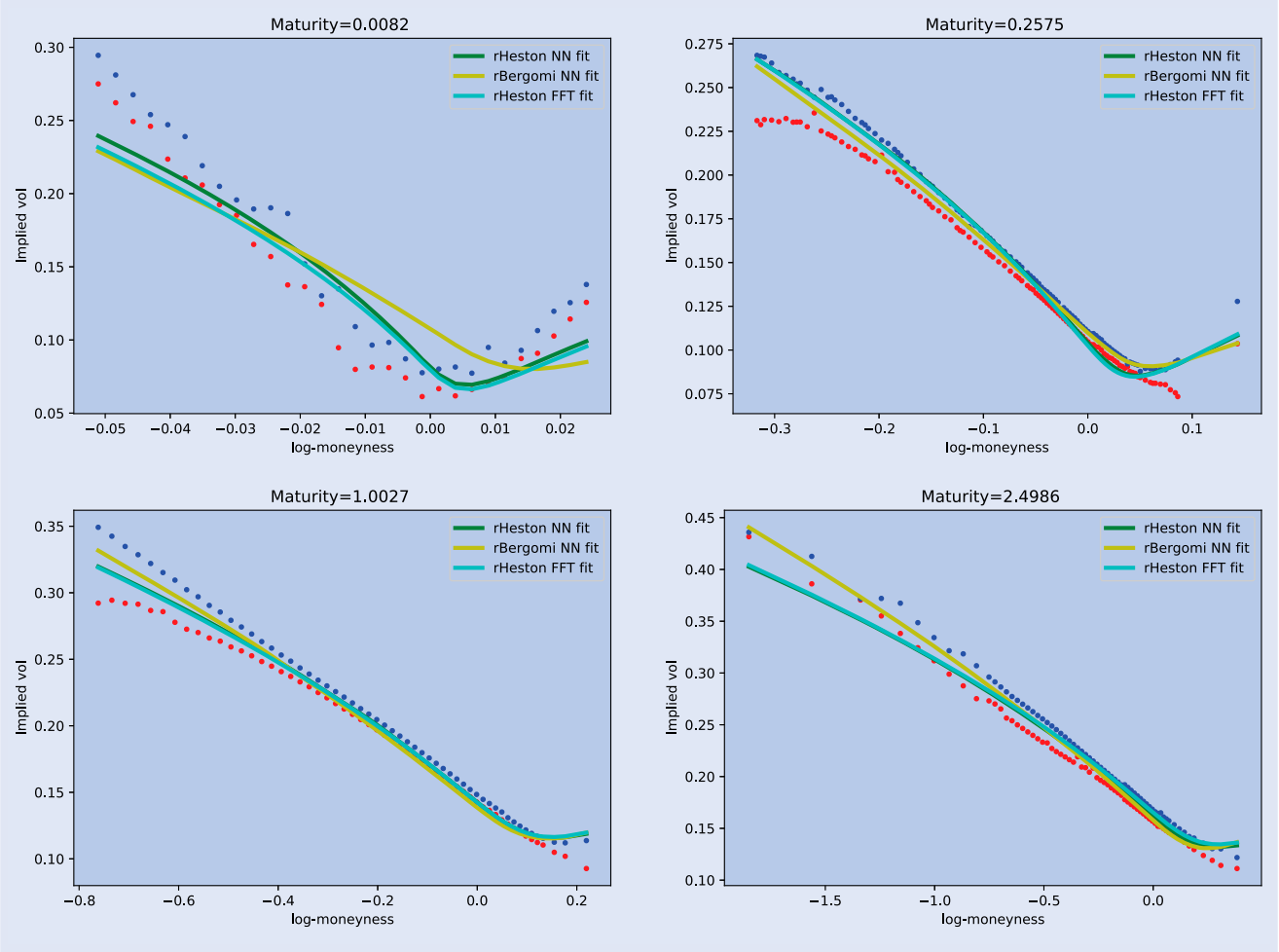


Figure 20. rBergomi (yellow) and rHeston fits as of June 18, 2014. Calibration via neural networks prescribes a parametric forward variance curve as in equation (4) to be calibrated together with model parameters. FFT pricing pre-computes a piecewise constant forward variance curve and makes it a state variable. Those correspond to the green curve and the cyan curve respectively, for the rHeston model.

5.1. Calibration of rough volatility models

Once we have checked that the neural network learnt the true pricing function and using a parametric forward variance curve is consistent with standard piecewise constant specifications, the last step to the validation of our pricing tool is to confront it with a completely different approach which comes after El Euch *et al.* (2019). As we already mentioned, the authors suggest that the forward variance curve is estimated from option prices and used as a state variable in a subsequent calibration procedure which only involves model parameters. No network can be advocated with this approach and availability of a closed-form solution for the characteristic function is therefore essential to prevent calibration times to explode. If such an alternative is available, however, neural network approaches should clearly confront with it, which fact is often times forgotten in the literature.

Preliminary construction of the forward variance curve requires evaluation of variance swaps via integration of option prices over a continuum of strikes.† Such an integration is usually performed on the market maturities only, and a piecewise constant forward variance curve obtained by differentiation. Fixing the resulting curve allows for separate calibration of the model parameters via the FFT. We do that in figure 20 and prove that our pointwise approach yields very similar results.

Still, the methodology by El Euch *et al.* (2019) may occasionally fail to provide a good description of the market data at very short times (say the first maturity in the market, T_1). This is indeed a consequence of the fact that what one is estimating with the log-strip is the value of the forward variance curve at time T_1, but no information is gathered about its previous history. Extrapolating the curve flat may therefore result in an over/under-estimation. In this sense, the advantage of a neural network approach is that one is encoding a whole piece of the curve over the period $[0, T_1]$ in the option price for maturity T_1. That's why the level of the smile in the top left corner of figure 21 is correct when using the pointwise approach with a parametric forward variance curve and a little bit biased when the curve is taken as a state variable.

Because extrapolation methods know nothing about the market prior to maturity T_1, encapsulating the primitive history of the curve into a neural network is the only way to guarantee a sensible fit to the shortest maturity.

Interpolation being safe, however, we can evaluate variance swaps on a much finer grid than the one imposed by the market and make the steps in the forward variance curve so short

† This is made possible by the tools from Vola Dynamics.

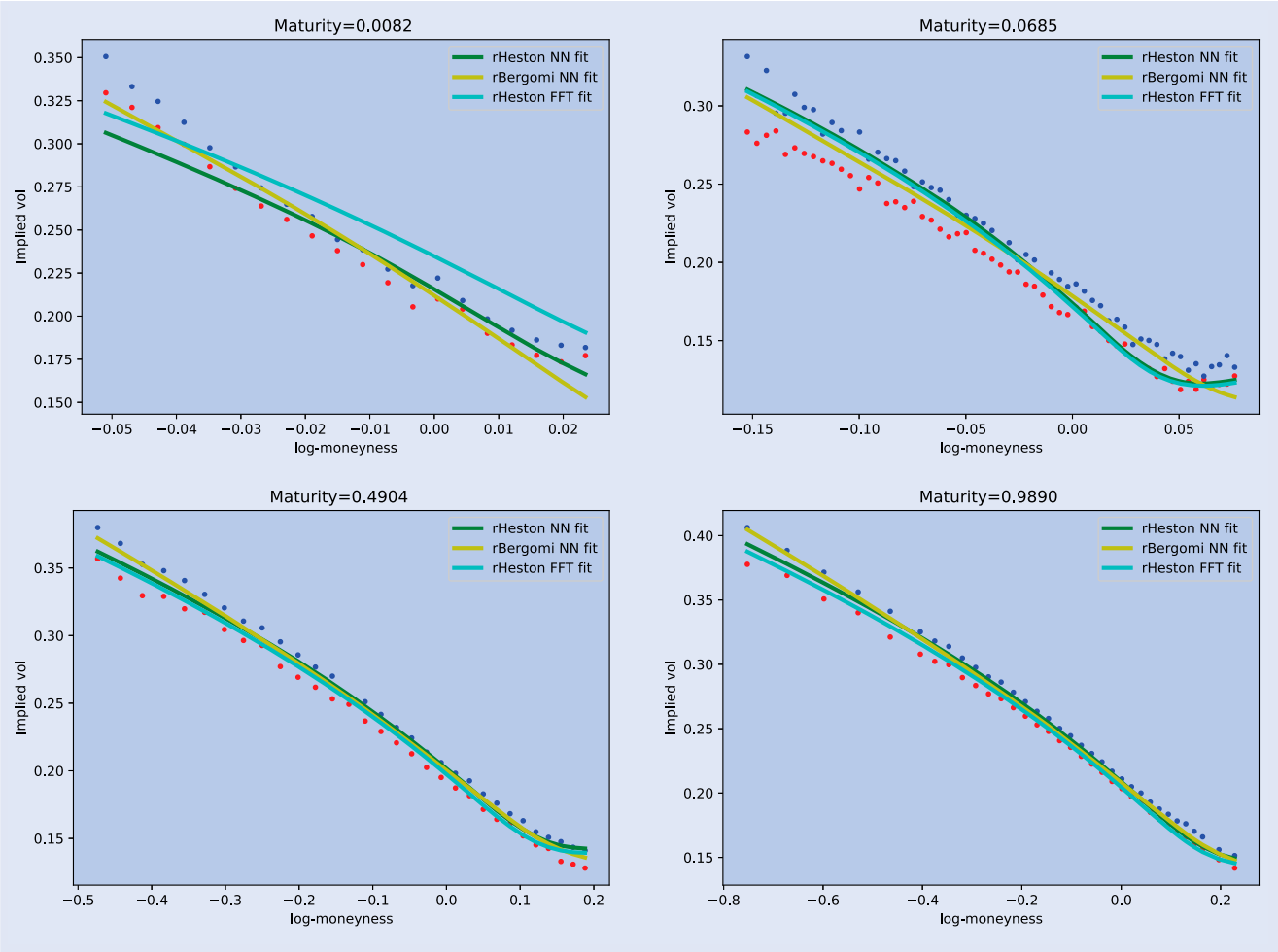


Figure 21. rBergomi (yellow) and rHeston fits as of March 22, 2011. Calibration via neural networks prescribes a parametric forward variance curve as in equation (4) to be calibrated together with model parameters. FFT pricing pre-computes a piecewise constant forward variance curve and makes it a state variable. Those correspond to the green curve and the cyan curve respectively, for the rHeston model.

that we get a sense of it in continuous time. We do so for a few surfaces in figure 22 and recognize that this is indeed going in the direction of the parametric form of equation (4).

As for calibration times under the pointwise approach, we had maybe better look at a piecewise constant specification of the forward variance curve first. Valuation of one single option by the neural network takes about 7e-06 seconds under this setup, which fact results in the following summary statistics in table 6 for calibration of entire volatility surfaces in our sample.

Making the forward variance curve parametric does not necessarily reduce calibration time. It will certainly be true that evaluation of option prices is slightly faster as the neural network is smaller (because the input layer is) but the overall calibration time depends on how many evaluations are needed for convergence. In particular, the problem with equation (4) is that there are regions in parameter space where the curve experiences very little change in response to changing the input. Then, the optimizer is likely to spend some time before exiting these regions if it ever visits them because the objective function is very flat here and steps very short. Luckily enough those regions are small and they correspond to values of β_1 and/or β_2 which are close to zero. That is why we say that numerical aspects are also to be taken into account when conceiving some parametrization of the forward variance curve. Nevertheless, calibration times with a parametric forward variance curve are very much in line with what Rømer (2022) achieves by working on dense adapted grids plus interpolation/extrapolation.

Furthermore, reducing the number of parameters makes it easier for the network to learn the underlying pricing function and helps in producing those very nice plots that we see in figures 18 and 19.

6. Conclusions

We propose a pointwise approach which is trained on volatility points coming from random grids. Generating entire surfaces in the form of random grids is indeed very convenient from a computational viewpoint, because the cost for one entire smile at time T is the same as a single strike for the same maturity. Most importantly, then, the method proves to be very competitive. It learns the underlying pricing function extremely well and it allows for calibration to the market in short time (less than one second on average). The random grid being adaptive in nature, our pointwise approach is financially sound and does not depend on any interpolation/extrapolation algorithm. Its use as a pricer is

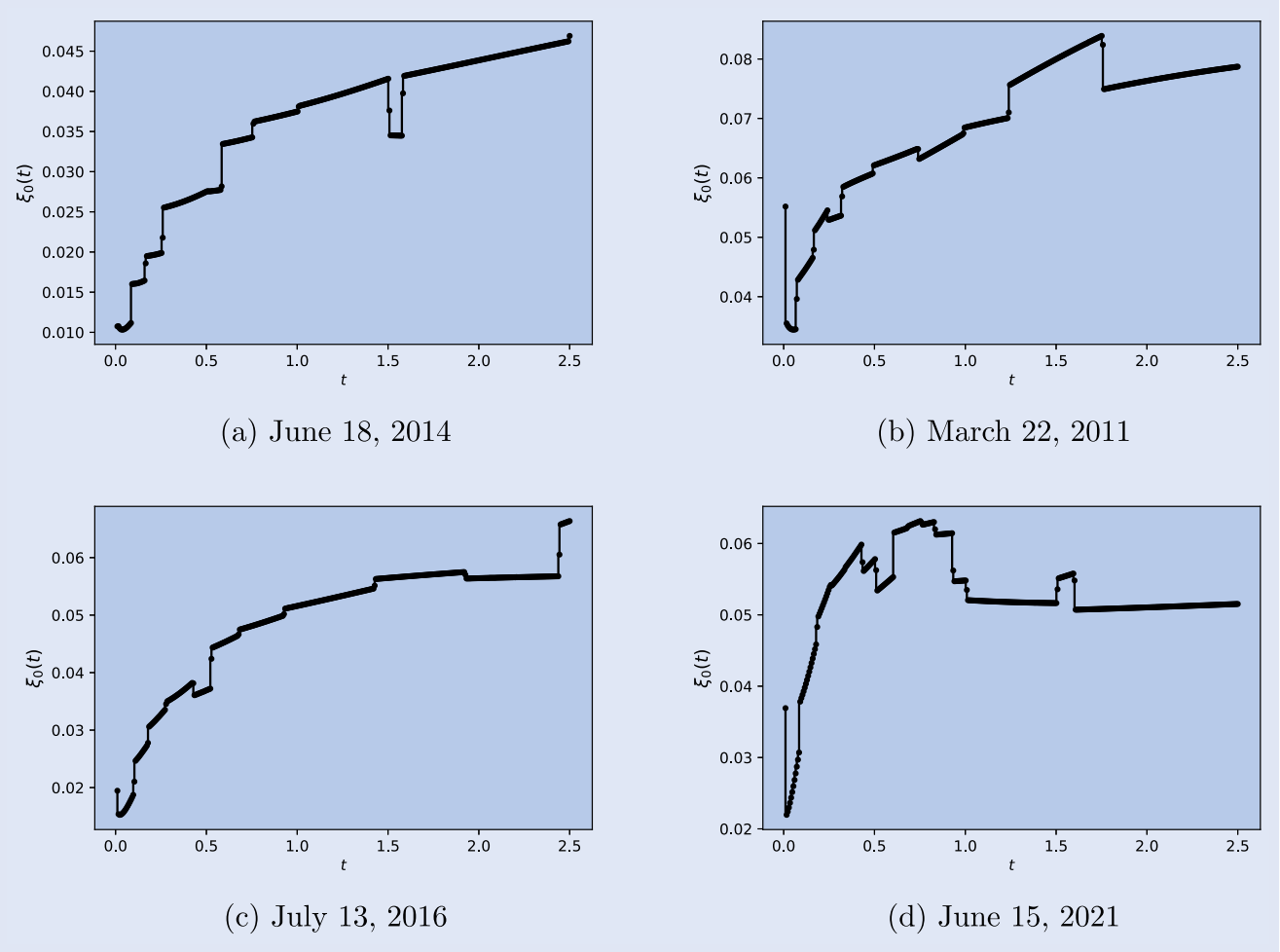


Figure 22. Market forward variance curve from applying log-strip techniques on a fine grid of market maturities as interpolated with Vola Dynamics. (a) June 18, 2014. (b) March 22, 2011. (c) July 13, 2016 and (d) June 15, 2021.

Table 6. Summary statistics for calibration time (in seconds) in our sample data, both under rHeston and rBergomi with piecewise constant and parametric forward variance curve.

	rHeston calib time		rBergomi calib time	
Piecewise constant fvc	avg	0.5852s	avg	0.7698s
	min	0.2266s	min	0.3836s
	max	1.4989s	max	1.7924s
Parametric fvc [equation (4)]	avg	0.8673s	avg	1.1438s
	min	0.2251s	min	0.2844s
	max	3.2470s	max	4.8668s

also possible, but requires some care. After performing the calibration step, the optimal parameters can be used by the trader to instantaneously recover via NN market consistent option prices for any desired contract specifications—as long as interior to the training set. Possible arbitrage opportunities may arise if the quality of the NN approximation of the true pricing function is poor. However, it is worth to say that we never experienced such a situation in all explorations presented in this paper.

Our work fits into the flourishing literature about neural network calibration of stochastic volatility models and provides a fresh perspective on the subject. We give new life to the pointwise approach and show how this can easily compete with grid-based methods that are now more widespread in the literature. Actually, we believe (and we show in the paper) that independence on whatever interpolation technique is a valuable aspect of the pointwise approach, that neural networks should retain.

In addition to this, we recognize that—once the optimal parameters have been calibrated to market data—extrapolation of the volatility surface to very short maturities can be achieved with the neural network in a way which is naturally consistent with the dynamical properties of the model.

We prove that our solution is providing robust estimation of model parameters in a controlled environment and test it against the market as well.

As far as rough volatility models are concerned, our contribution also embraces a novel parametrization of the forward variance curve which guarantees very good fits to the market

while reducing the dimensionality of the calibration process and easing the learning task. We also come back to a treatment of the forward variance curve as a state variable and make it 'continuous' via interpolation with the tools from Vola Dynamics LCC. When we do so, we observe that the shapes that we recover are very much in line with our parametric form, thus further corroborating it.

The domain of our neural network pricer obviously extends to non-rough models. More specifically, we have in mind potential application of the method to PDV models for which option prices are only known after simulation. What is particularly interesting with those model is in fact the possibility for pricing options on the index and the VIX, thus using neural networks for the joint calibration problem. Also, because the neural network seems to have learnt the true pricing function very well, we can use it for construction of the Dupire function to be used with local stochastic volatility models.

Open Scholarship

This article has earned the Center for Open Science badge for Open Materials. The materials are openly accessible at https://github.com/fabioBaschetti/random-grid-NN-calib.

Acknowledgments

We warmly thank Vola Dynamics LLC for making their tools available for academic purposes. Many explorations in the paper would not have been possible without their support.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Fabio Baschetti http://orcid.org/0000-0001-5973-4437
Giacomo Bormetti http://orcid.org/0000-0003-3542-2505
Pietro Rossi http://orcid.org/0000-0002-2414-3458

References

Abi Jaber, E., Lifting the Heston model. *Quant. Finance*, 2019, **19**(12), 1995–2013.
Abi Jaber, E. and El Euch, O., Multifactor approximation of rough volatility models. *SIAM J. Financ. Math.*, 2019, **10**(2), 309–349.
Baschetti, F., Bormetti, G., Romagnoli, S. and Rossi, P., The SINC way: A fast and accurate approach to Fourier pricing. *Quant. Finance*, 2022, **22**(3), 427–446.
Bayer, C., Friz, P. and Gatheral, J., Pricing under rough volatility. *Quant. Finance*, 2016, **16**(6), 887–904.
Bayer, C. and Stemper, B., Deep calibration of rough stochastic volatility models. Working Paper, arXiv:1810.03399, 2018.
Bennedsen, M., Lunde, A. and Pakkanen, M.S., Hybrid scheme for Brownian semistationary processes. *Finance Stoch.*, 2017, **21**(4), 931–965.
Blanc, P., Donier, J. and Bouchaud, J.P., Quadratic Hawkes processes for financial prices. *Quant. Finance*, 2017, **17**(2), 171–188.
Bourgey, F., De Marco, S., Friz, P.K. and Pigato, P., Local volatility under rough volatility. *Math. Finance*, 2023, **33**(4), 1119–1145.
Callegaro, G., Grasselli, M. and Pages, G., Fast hybrid schemes for fractional Riccati equations (rough is not so tough). *Math. Oper. Res.*, 2021, **46**(1), 221–254.
Cherubini, U., Della Lunga, G., Mulinacci, S. and Rossi, P., *Fourier Transform Methods in Finance*, 2010 (John Wiley & Sons Inc: Chichester).
Dall'Acqua, E., Longoni, R. and Pallavicini, A., Rough–Heston local-volatility model. *Int. J. Theor. Appl. Finance*, 2023. http://dx.doi.org/10.1142/S0219024923500218
Diethelm, K., Ford, N.J. and Freed, A.D., Detailed error analysis for a fractional Adams method. *Numer. Algorithms*, 2004, **36**(1), 31–52.
El Euch, O., Gatheral, J. and Rosenbaum, M., Roughening Heston. *Risk*, 2019, **May**, 84–89.
El Euch, O. and Rosenbaum, M., The characteristic function of rough Heston models. *Math. Finance*, 2019, **29**(1), 3–38.
Euch, O.E. and Rosenbaum, M., Perfect hedging in rough Heston models. *Ann. Appl. Probab.*, 2018, **28**(6), 3813–3856.
Fukasawa, M., Short-time at-the-money skew and rough fractional volatility. *Quant. Finance*, 2017, **17**(2), 189–198.
Gatheral, J., *The Volatility Surface: A Practitioner's Guide*, 2011 (John Wiley & Sons: Hoboken, NJ).
Gatheral, J., Jusselin, P. and Rosenbaum, M., The quadratic rough Heston model and the joint S&P 500/VIX smile calibration problem. *Risk*, 2020, **May**.
Gatheral, J. and Radoicic, R., Rational approximation of the rough Heston solution. *Int. J. Theor. Appl. Finance*, 2019, **22**(3), 1950010.
Guyon, J., Path-dependent volatility. *Risk*, 2014, **October**.
Guyon, J. and Lekeufack, J., Volatility is (mostly) path-dependent. *Quant. Finance*, 2023, **23**(9), 1221–1258.
Hernandez, A., Model calibration with neural networks. *Risk*, 2017, **June**.
Horvath, B., Muguruza, A. and Tomas, M., Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models. *Quant. Finance*, 2021, **21**(1), 11–27.
Itkin, A., Deep learning calibration of option pricing models: Some pitfalls and solutions. *Risk*, 2020, **April**.
Liu, S., Borovykh, A., Grzelak, L.A. and Oosterlee, C.W., A neural network-based framework for financial model calibration. *J. Math. Ind.*, 2019, **9**(1), 1–28.
McCrickerd, R. and Pakkanen, M.S., Turbocharging Monte Carlo pricing for the rough Bergomi model. *Quant. Finance*, 2018, **18**(11), 1877–1886.
Nelson, C.R. and Siegel, A.F., Parsimonious modeling of yield curves. *J. Bus.*, 1987, **60**, 473–489.
Parent, L., The EWMA Heston model. *Quant. Finance*, 2023, **23**(1), 71–93.
Rømer, S.E., Empirical analysis of rough and classical stochastic volatility models to the SPX and VIX markets. *Quant. Finance*, 2022, **22**(10), 1805–1838.

Appendices

Appendix 1

We report summary statistics for the estimation of rHeston model parameters under different specifications of the network used for calibration (fxd, ada, pnt for fixed grid, adapted grid, and pointwise approach, respectively). Each of the parameters has its own

Table A1. Summary statistics for the error associated with the estimation of the roughness parameter H in a controlled environment using different network specifications: fixed grid (fxd), adapted grid (ada) and pointwise (pnt) [with points on the adapted grid]. Networks have piecewise constant forward variance curve, the calibration data has flat.

Summary statistics e_H true vs ...	fxd	ada	pnt
min	−0.1219	−0.0353	−0.0200
mean	0.0031	−0.0003	0.0015
med	0.0020	−0.0009	0.0004
q95	0.0266	0.0089	0.0075
max	0.0697	0.0386	0.0622
std	0.0142	0.0057	0.0050
overestimation ratio	0.225	0.614	0.406
2-tail mass at 0.02	0.106	0.012	0.013
2-tail mass at 0.03	0.054	0.005	0.006
2-tail mass at 0.05	0.019	0.000	0.002

Table A2. Summary statistics for the error associated with the estimation of the VOV ν in a controlled environment using different network specifications: fixed grid (fxd), adapted grid (ada) and pointwise (pnt) [with points on the adapted grid]. Networks have piecewise constant forward variance curve, the calibration data has flat.

Summary statistics e_ν true vs ...	fxd	ada	pnt
min	−0.0302	−0.0413	−0.0049
mean	0.0065	−0.0008	0.0075
med	0.0016	−0.0038	0.0047
q95	0.0419	0.0196	0.0201
max	0.1625	0.2542	0.2119
std	0.0176	0.0174	0.0126
overestimation ratio	0.260	0.808	0.007
2-tail mass at 0.02	0.118	0.066	0.051
2-tail mass at 0.03	0.079	0.043	0.035
2-tail mass at 0.05	0.035	0.028	0.017

dedicated table, where we keep track of the following quantities: largest negative error (min), average error (mean), median error (med), 95th percentile of the distribution of the errors (q95), largest positive error (max), standard deviation of the distribution of the errors (std), fraction of the runs which result in the true parameter being overestimated by the network (overestimation ratio), fraction of the runs which result in an absolute error $|e.| > x$ (2-tail mass at x).

Information in the tables below provides numerical support for proper interpretation of the figures in section 4.1.

Appendix 2. An additional robustness check

In the main text, we suggest a methodology to replace any model pricing function, semi-analytical (e.g. rHeston) or entirely Monte Carlo (e.g. rBergomi), with a Neural Network approximation. The issue of possible absence of arbitrage violations pertains to the model specification. If the pricing model allows for arbitrages, then the NN will learn a badly specified model and will suffer the same shortcomings. If the pricing model is an arbitrage-free model—as for the rHeston and rBergomi models—the NN will learn arbitrage-free pricing functions. This appendix provides an additional robustness check that the NN properly learns the arbitrage-free pricing functions. This is purely a sanity check provided in developing the model. In no way we do suggest that one would need to run it after any calibration.

Table A3. Summary statistics for the error associated with the estimation of the correlation ρ in a controlled environment using different network specifications: fixed grid (fxd), adapted grid (ada) and pointwise (pnt) [with points on the adapted grid]. Networks have piecewise constant forward variance curve, the calibration data has flat.

Summary statistics e_ρ true vs ...	fxd	ada	pnt
min	−0.0301	−0.1023	−0.0138
mean	0.0025	−0.0013	0.0023
med	0.0003	−0.0029	0.0007
q95	0.0120	0.0194	0.0140
max	0.1870	0.1779	0.1041
std	0.0139	0.0201	0.0076
overestimation ratio	0.406	0.777	0.343
2-tail mass at 0.02	0.039	0.104	0.034
2-tail mass at 0.03	0.023	0.066	0.016
2-tail mass at 0.05	0.016	0.037	0.002

Sufficient conditions for absence of arbitrage in call option prices $C = C(F, K, T)$ are well known to be given as follows

$$\frac{\partial C}{\partial T} > 0, \quad \frac{\partial C}{\partial K} < 0, \quad \frac{\partial^2 C}{\partial K^2} > 0. \tag{A1}$$

A discrete version of these is straightforward to derive and can be found in Itkin (2020). The idea is that one fixes the option maturity T and the forward price F, and looks at call options $C(K_1), C(K_2), C(K_3)$ for strikes $K_1 < K_2 < K_3$. Then

$$C(K_3) > 0, \quad C(K_2) > C(K_3)$$

imposes no vertical spread, and

$$(K_3 - K_2)C(K_1) - (K_3 - K_1)C(K_2) + (K_2 - K_1)C(K_3) > 0$$

guarantees the absence of any butterfly spread arbitrage. Fixing the strike and allowing the option maturity to move, we verify the absence of calendar spread arbitrage as

$$C(T_2) > C(T_1)$$

for maturities $T_1 < T_2$.

We calibrate to about 50 daily market volatility surfaces and store the optimal parameter values. Using the NN, we compute the implied volatility surface for each day—and corresponding parameter set—over an extremely dense strike and time-to-maturity grid. We consider time-to-maturities running from 2 days up to 2.5 years ($dt = \frac{1}{365}$) with a daily resolution and evenly sampled strikes ($dK = 0.01$) over the range $(S_0(1 - 0.55\sqrt{t}), S_0(1 + 0.30\sqrt{t}))$. On the corresponding price surfaces, computed via the Black-Scholes formula, we look for violations of the arbitrage conditions. With a parametric forward variance curve—and the dimensionality of the learning problem consequently reduced—we did not detect any violation out of more than 80.000 sampled points.