On deep calibration of (rough) stochastic volatility models

Christian Bayer
TU Berlin and WIAS
christian.bayer@wias-berlin.de

Blanka Horvath
Department of Mathematics, King's College London
blanka.horvath@kcl.ac.uk, b.horvath@imperial.ac.uk

Aitor Muguruza
Department of Mathematics, Imperial College London & NATIXIS
aitor.muguruza-gonzalez15@imperial.ac.uk

Benjamin Stemper
TU Berlin and WIAS
benjamin.stemper@wias-berlin.de

Mehdi Tomas
CMAP & LadHyx, École Polytechnique
mehdi.tomas@polytechnique.edu

August 26, 2019

Abstract

Techniques from deep learning play a more and more important role for the important task of calibration of financial models. The pioneering paper by Hernandez [Risk, 2017] was a catalyst for resurfacing interest in research in this area. In this paper we advocate an alternative (two-step) approach using deep learning techniques solely to learn the pricing map – from model parameters to prices or implied volatilities – rather than directly the calibrated model parameters as a function of observed market data. Having a fast and accurate neural-network-based approximating pricing map (first step), we can then (second step) use traditional model calibration algorithms. In this work we showcase a direct comparison of different potential approaches to the learning stage and present algorithms that provide a sufficient accuracy for practical use. We provide a first neural network-based calibration method for rough volatility

The authors are grateful to Ben Wood, Jim Gatheral and Ryan McCrickerd for stimulating discussions. MT acknowledges financial support from the Econophysique et Systmes Complexes chair under the aegis of the Fondation du Risque, a joint initiative by the Fondation de lÉcole Polytechnique, lÉcole Polytechnique and Capital Fund Management. CB and BS are grateful for financial support by the DFG through research grants BA5484/1 and FR2943/2. The present paper combines and consolidates findings of its two predecessor papers, [7] and [35].

models for which calibration can be done on the fly. We demonstrate the method via a hands-on calibration engine on the rough Bergomi model, for which classical calibration techniques are difficult to apply due to the high cost of all known numerical pricing methods. Furthermore, we display and compare different types of sampling and training methods and elaborate on their advantages under different objectives. As a further application we use the fast pricing method for a Bayesian analysis of the calibrated model.

2010 *Mathematics Subject Classification*: 60G15, 60G22, 91G20, 91G60, 91B25
Keywords: Rough volatility, volatility modelling, Volterra process, machine learning, accurate price approximation, calibration, model assessment, Monte Carlo

Contents

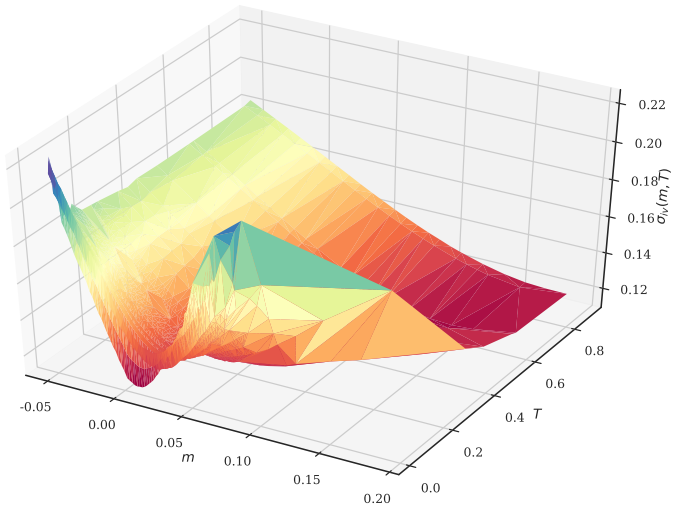


Figure 1: **SPX Market Implied Volatility surface on 15th February 2018.** IVs have been inverted from SPX Weekly European plain vanilla call mid prices and the interpolation is a (non-arbitrage-free) Delaunay triangulation. Axes denote log-moneyness $m = \log(K/S_0)$ for strike K and spot S_0, time to maturity T in years and market implied volatility $\sigma_{\text{iv}}(m, T)$.

1 Introduction

Almost half a century after its publication, the option pricing model by Black, Scholes and Merton remains one of the most popular analytical frameworks for pricing and hedging European options in financial markets. A part of its success stems from the availability of explicit and hence instantaneously computable closed formulas for both theoretical option prices and option price sensitivities to input parameters (*Greeks*), albeit at the expense of assuming that *volatility* – the standard deviation of log returns of the underlying asset price – is deterministic and constant. Still, in financial practice, the Black-Scholes model is often considered a sophisticated transform between option prices and Black-Scholes (BS) *implied volatility (IV)* σ_{iv} where the latter is defined as the constant volatility input needed in the BS formula to match a given (market) price. It is a well-known fact that in empirical IV surfaces obtained by transforming market prices of European options to IVs, it can be observed that IVs vary across moneyness and maturities, exhibiting well-known smiles and at-the-money (ATM) skews and thereby contradicting the flat surface predicted by Black-Scholes (Figure 1). In particular, Bayer, Friz, and Gatheral [5] report empirical at-the-money volatility skews of the form

$$\left| \frac{\partial}{\partial m} \sigma_{\text{iv}}(m, T) \right| \sim T^{-0.4}, \quad T \to 0, \tag{1}$$

for log moneyness m and time to maturity T.

While plain vanilla European call and put options often show enough liquidity to be marked-to-market, pricing and hedging path-dependent options (so-called *exotics*) necessitates an option

pricing model that prices European options *consistently* with respect to observed market IVs across moneyness and maturities. In other words, it should parsimoniously capture stylized facts of empirical IV surfaces. To address the shortcomings of Black-Scholes and incorporate the stochastic nature of volatility itself, popular bivariate diffusion models such as SABR [27] or Heston [29] have been developed to capture *some* important stylized facts. However, according to Gatheral [23], diffusive stochastic volatility models in general fail to recover the exploding power-law nature (1) of the volatility skew as time to maturity goes to 0 and instead predict a constant behaviour.

Sparked by the seminal work of [1, 22, 24], we have since seen a shift from classical diffusive modeling towards so-called rough stochastic volatility models. They may be defined as a class of *continuous-path* stochastic volatility models where the instantaneous volatility is driven by a stochastic process with Hölder regularity smaller than Brownian Motion, typically modeled by a fractional Brownian Motion with Hurst parameter $H < \frac{1}{2}$. The evidence for this paradigm shift is by now overwhelming, both under the physical measure where time series analysis suggests that log realized volatility has Hölder regularity in the order of ≈ 0.1 [8, 24] and also under the pricing measure where the empirically observed power-law behaviour of the volatility skew near zero may be reproduced in the model [1, 5, 6, 22]. Serious computational and mathematical challenges arise from the non-Markovianity of fractional Brownian motion, effectively forcing researchers to resort to asymptotic expansions [6, 17] in limiting regimes or (variance-reduced) Monte Carlo schemes [4, 5, 34, 47] to compute fair option prices. This poses considerable bottlenecks for calibration of rough volatility models for practical purposes. One contribution of this work is to provide and explore different neural network based solutions to the task of fast calibration of rough volatility models.

The solution we provide here is demonstrated on the rough Bergomi model but due to the nature of neural network approximations (as opposed to static polynomial approximations) it is fundamentally model agnostic and it consistently[1] carries over to other rough volatility models (of the same complexity) and to classical stochastic volatility models, which are by nature simpler to approximate.

The "need for speed" is by no means limited to rough volatility models, although our initial motivation was indeed the rough Bergomi model. Parallel to this work, Ferguson and Green address in [19, Section 1.1] the ongoing struggle for faster pricing algorithms for more and more complex products and propose a deep learning approach to pricing basket options in a lognormal setting to achieve considerable speed-ups over Monte Carlo pricers. High dimensional problems as in [19] are one useful applications of the speedup resuliting from this methodology. But it can also enable us to speed up more involved numerical methods for benchmark stochastic volatility models: multiple integrals [2], Monte Carlo-type methods [48] or Finite Element Methods [36] for the SABR model can thus compete in speed with the original SABR expansion formula [27], by pre-learning them through the DNN. In related contexts deep BSDE solvers have been used to replace Monte Carlo methods for solving Backward Stochastic Differential Equations in high dimension [28, 33, 49] which can arise from pricing problem. Other authors used computational speedups provided by neural networks in the context of computationally expensive valuation adjustments [26, 33].

[1] By consistency we mean here that the proposed network (with the same architecture) can be trained on different models consistently without further modifications and yield satisfactory results irrespective of the chosen model for training. Our numerical experiments show that for the calibration of classical stochastic volatility models (SABR, Heston) a simpler network architecture is sufficient, while rough volatility models require a more nuanced network design.

The work we present here is very much in the spirit of the pioneering work of Avellaneda, Carelli and Stella [3]. Our focus in this work is on model calibration of stochastic volatiliy models and we propose computationally efficient and ready-to-use algorithms that can be applied to a variety of settings. Bearing in mind that deep neural network solutions are often challenged by concerns of generalisation and "black-box-solutions", our goal is to limit the application of neural networks to parts of the calibration process that we can control and validate. As a first step, we identify the parts of the calibration process that are mainly responsible for the prevailing calibration bottlenecks, which we will replace by a deep neural network. To motivate our approach, recall that model calibration is the optimization procedure of finding model parameters such that the IV surface induced by the model best approximates a given market IV surface in an appropriate metric. In the absence of an analytical solution, it is standard practice to solve the arising weighted non-linear least squares problem using iterative optimizers such as Levenberg-Marquardt (LM) [43, 44]. However, these optimizers rely on the repetitive evaluation of the function φ from the space of model & option parameters (and external market information) to model BS implied volatility. If each such evaluation involves a time– and/or memory–intensive operation such as a Monte Carlo simulation in the case of *rough Bergomi* [5] or other (rough) stochastic volatility models, this makes efficient calibration prohibitively expensive.

To bridge this computational gap and motivated by their prowess in approximating smooth functions [32], neural networks have been used to build fast solutions to the calibration problem. Unsurprisingly, the tremendous rise in popularity of *Deep learning* among academics and practitioners in recent years is closely tied to the widespread availability of cheap, high performance computing hardware as well as to theoretical advancements. Fundamentally, most of the solutions in a calibration context build on the capability of multi-layered artificial neural networks to closely approximate functions f only implicitly available through *labeled datasets* of input-output pairs $\{(x_i, f(x_i))\}_{i=1}^N$.

In this context, we distinguish two kinds of approaches. The first, pioneered by Hernandez [30], seeks to learn the mapping from implied volatility surfaces to model parameters (inverse problem) directly. In [30], Hernandez proposes to use a neural network to learn the complete calibration routine taking market data as inputs and returning calibrated model parameters, and calibrates the popular short rate model of Hull and White [37] to market data in numerical experiments. In Section 3.1 we describe this approach in more detail and perform a similar calibration experiment with the Heston Model. In the rest of this paper, we will refer to it as the *one-step* approach. In a second strand of research neural networks have been applied not directly to calibration problems, but simply to the obtain an approximative representation of derivative valuations, i.e. of option pricing maps: For example Hutchinson, Lo and Poggio [38] such as Culkin and Das [12] applied neural networks to learn the Black-Scholes formula and McGhee demonstrates in [46] a neural networks representation of the lognormal SABR model. In this paper we explore the advantages of shaping this second strand of research into a building block of a single *two-step approach*.

The *two-step* approach, which we highlight in this paper, first approximates the pricing map, (denoted, by φ from model parameters to option prices) by a neural network (Step **(i)**) before calibrating the model, (via traditional calibration algorithms applied to the approximate pricing map φ_{NN}) to market data (Step **(ii)**). Thereby we optimally leverage the capability of neural networks to approximate functions which are only implicitly available through input-output pairs

$\{(x_i, \phi(x_i))\}_{i=1}^N$, by training a fully-connected neural network on specifically tailored, synthetically generated training data to learn an approximative representation φ_{NN} of the pricing functional φ. Details of this approach and its benefits are further explained in Section 3.

There are different ways of approximating the pricing map (Step **(i)**): One could consider a *pointwise* approach where strikes and maturities are input parameters of the pricing map along model parameters. Alternatively to this we also explore the advantages of proceeding here instead with a *gridwise* approach, by first setting strikes and maturities before learning the map from model parameters to the implied volatility surface (with corresponding strikes and maturities). In this work we showcase a direct comparison these approaches to the learning stage (Step **(i)**) and present algorithms that provide a sufficient accuracy for practical use, but are computationally efficient enough for daily practice on a large scale:

In particular, in Section 3.2 we compare different network architectures and sampling methods according to different modelling objectives. Among these, the grid-based approach is particularly designed for applicability and efficiency in every day calibration practice. The novelty of our grid-based approach will allow us to tackle the calibration problem with a remarkably small neural network (3 layers 30 neurons), which to the best of our knowledge is the smallest network in the literature to successfully solve the calibration/pricing task. As opposed to the aforementioned works in the literature, we do not resort to GPU's or heavy computational resources, since the architecture of the problem easily permits to run the code on a standard CPU. This in turn, opens the door to its practical implementation in the financial industry without the need to update current hardware systems.

The overall benefits of the *two-step approach* are plentiful:

- First, evaluations of φ_{NN} amount to cheap and almost instantaneous forward runs of a pre-trained network. Second, automatic differentiation of φ_{NN} with respect to the model parameters returns fast and accurate approximations of the Jacobians needed for the LM calibration routine. Used together, they allow for the efficient calibration of *any* (rough) stochastic volatility model including *rough Bergomi*.
- The *two-step* approach also has overwhelming risk management benefits. Firstly, we can understand and interpret the output of our neural network and therefore test the output as a function of model parameters against traditional numerical methods. (Indeed, the output values correspond to option prices in the model under consideration.) The second overwhelming advantage is that existing risk management libraries of models remain valid with minimal modification. The neural network is only used as a computational enhancement of models, and therefore, the knowledge and intuition gathered in many years of experience with traditional models remains useful.
- The training becomes more robust (with respect to generalisation errors on unseen data). Additionally, the trained network is independent from market data, and, in particular, from changing market environments.
- We can train the network to synthetic data – model prices or implied volatilities computed by any adequate numerical method. In particular, we can easily provide as large training sets as desired.

Both generating the synthetic data set as well as the actual neural network training are expensive in time and computing resource requirements, yet they only have to be performed a single time. Trained networks may then be quickly and efficiently saved, moved and deployed. We demonstrate this first advantage in a further application: a Bayesian calibration experiment, which is facilitated by our ability to nearly instantaneously call functional evaluations of option prices in a given model. To quantify the uncertainty about model parameter estimates obtained by calibrating with φ_{NN}, we infer model parameters in a Bayesian spirit from (i) a synthetically generated IV surface and (ii) SPX market IV data. In both experiments, a simple (weighted) Bayesian nonlinear regression returns a (joint) posterior distribution over model parameters that (1) correctly identifies sensible model parameter regions and (2) places its peak at or close to the true (in the case of the synthetic IV) or previously reported [5] (in the case of the SPX surface) model parameter values. Both experiments thus confirm the idea that φ_{NN} is sufficiently accurate for calibration.

The paper is organised as follows: In Section 2 we present an abstract point of view on model calibration in finance. In Section 3 we give an overview of applications of techniques from deep learning to model calibration. We also introduce our own framework and discuss possible advantages and disadvantages as compared to other approaches. In Section 4 we focus on the concrete implementation of our methods, both for the learning and for the actual calibration stage. Numerical experiments are then presented in Section 5. In addition, we also apply the network in a Bayesian approach. The Appendix A contains a numerical comparison with an alternative deep learning approach to calibration.

2 Model calibration

Calibration describes the procedure of tuning model parameters to fit a model surface to an empirical implied volatility surface obtained by transforming liquid European option market prices to Black-Scholes implied volatilities. A mathematically convenient approach consists of minimizing the weighted squared differences between market and model implied volatlities of $N \in \mathbb{N}$ plain vanilla European options.

Suppose that a model is parametrized by a set of parameters Θ, i.e., by $\theta \in \Theta$. We refer to Example 1 for a concrete example. Furthermore, we consider options parametrized by a parameter $\zeta \in Z$. E.g., for put and call options we generally have $\zeta = (T, k)$, the option's maturity and log-moneyness. There might be further parameters which are needed to compute prices but can be observed on the market and, hence, do not need to be calibrated. For instance, the spot price of the underlying, the interest rate, or the forward variance curve in Bergomi-type models (see [9]) falls under this type. For this quick overview, we ignore this category. We introduce the *pricing map*

$$(\theta, \zeta) \mapsto P(\theta, \zeta),$$

the price of an option with parameters ζ in the model with parameters θ. We are also given market prices $\mathcal{P}(\zeta)$ for options parametrized by ζ for a (finite) subset $\zeta \in Z' \subset Z$ of all possible option parameters. *Calibration* now identifies a model parameter θ which minimizes a chosen distance δ between model prices $(P(\theta, \zeta))_{\zeta \in Z'}$ and market prices $(\mathcal{P}(\zeta))_{\zeta \in Z'}$, i.e.,

$$\widehat{\theta} = \operatorname*{argmin}_{\theta \in \Theta} \delta \left((P(\theta, \zeta))_{\zeta \in Z'}, (\mathcal{P}(\zeta))_{\zeta \in Z'} \right).$$

Remark 1. Financial practice often prefers to work with implied volatilities rather than option prices, and we will also do so in the numerical parts of this paper. For the purpose of this introduction, any mentioning of a *price* may be, mutatis mutandis, replaced by the corresponding implied volatility.

In fact, the most usual way to choice of a distance function δ is a suitably weighted least squares function, i.e.,

$$\widehat{\theta} = \operatorname*{argmin}_{\theta\in\Theta} \sum_{\zeta\in Z'} w_\zeta \left(P(\theta,\zeta) - \mathcal{P}(\zeta)\right)^2.$$

Here, the weights w_ζ can be chosen in order to reflect importance of an option at ζ and the reliability of the market observation $\mathcal{P}(\zeta)$. For instance, a reasonable choice might be the inverse of the bid-ask spread (see [11] for a motivation), which puts low weight on prices of illiquid options.

As long as the number of model parameters is smaller than the number $|Z'|$of calibration instruments, the calibration problem is an example of an overdetermined non-linear least squares problem, usually solved numerically using iterative solvers such as the de-facto standard Levenberg-Marquardt (LM) algorithm [43, 44]. Let $\boldsymbol{J} = \boldsymbol{J}(\theta)$ denote the Jacobian of the map $\theta \mapsto (P(\theta,\zeta)_{\zeta\in Z'}$ and let

$$\boldsymbol{R}(\theta) \coloneqq (P(\theta,\zeta) - \mathcal{P}(\zeta))_{\zeta\in Z'}$$

denote the residual, then the Levenberg-Marquart algorithm iteratively computes increments $\Delta\theta_k \coloneqq \theta_{k+1} - \theta_k$ by solving

$$\left[\boldsymbol{J}(\mu_k)^T \boldsymbol{W} \boldsymbol{J}(\mu_k) + \lambda \boldsymbol{I}\right] \Delta\theta_k = \boldsymbol{J}(\mu_k)^T \boldsymbol{W} \boldsymbol{R}(\mu_k) \tag{2}$$

where $\boldsymbol{I}$ denotes the identity matrix, $\boldsymbol{W} = \operatorname{diag}(w_\zeta)$, and $\lambda \in \mathbb{R}$.

Algorithm 1: Levenberg-Marquart calibration

Input: Implied vol map $\widetilde{\boldsymbol{P}}$ and its Jacobian $\widetilde{\boldsymbol{J}}$, market quotes $\boldsymbol{\mathcal{P}}$
Parameters: Lagrange multiplier $\lambda_0 > 0$, maximum number of iterations $n_{\max}$, minimum tolerance of step norm $\varepsilon_{\min}$, bounds $0 < \beta_0 < \beta_1 < 1$
Result: Calibrated model parameters $\theta^\star$

initialize model parameters $\theta = \theta_0$ and step counter $n = 0$;
compute $\widetilde{\boldsymbol{R}}(\theta) = \widetilde{\boldsymbol{P}}(\theta) - \boldsymbol{\mathcal{P}}$ and $\widetilde{\boldsymbol{J}}(\theta)$ and solve normal equations (2) for $\Delta\theta$;
while $n < n_{max}$ *and* $\|\Delta\theta\|_2 > \varepsilon$ **do**
  compute relative improvement $c_\theta = \frac{\left\|\widetilde{\boldsymbol{R}}(\theta)\right\|_2 - \left\|\widetilde{\boldsymbol{R}}(\theta+\Delta\theta)\right\|_2}{\left\|\widetilde{\boldsymbol{R}}(\theta)\right\|_2 - \left\|\widetilde{\boldsymbol{R}}(\theta)+\widetilde{\boldsymbol{J}}(\mu)\Delta\theta\right\|_2}$ with respect to predicted improvement under linear model;
  if $c_\theta \le \beta_0$ **then** reject $\Delta\theta$, set $\lambda = 2\lambda$;
  if $c_\theta \ge \beta_1$ **then** accept $\Delta\theta$, set $\theta = \theta + \Delta\theta$ and $\lambda = \frac{1}{2}\lambda$;
  compute $\widetilde{\boldsymbol{R}}(\theta)$ and $\widetilde{\boldsymbol{J}}(\theta)$ and solve normal equations (2) for $\Delta\theta$;
  set $n = n + 1$;
end

It is hence necessary that the *normal equations* (2) be quickly and accurately solved for the iterative step $\Delta\theta_k$. In a general (rough) stochastic volatility setting this is problematic: The true implied volatility map as well as its Jacobian $\boldsymbol{J}$ are unknown in analytical form. In the absence of an analytical expression for $\Delta\theta_k$, an immediate remedy is:

(I) Replace the (theoretical) true pricing (or implied volatility) map P by an efficient numerical approximation $\tilde{P}$ such as Monte Carlo, Fourier pricing.

(II) Apply finite-difference methods to $\tilde{P}$ to compute an approximate Jacobian $\tilde{\boldsymbol{J}}$.

In particular, in many (rough) stochastic volatility models such as the *rough Bergomi model* (see Example 1), expensive Monte Carlo simulations have to be used to approximate the pricing map. In a common calibration scenario where the normal equations (2) have to be solved frequently, the approach outlined above thus renders calibration prohibitively expensive.

Remark 2. We note that many modern tensor-based machine learning frameworks are ideally suited for calibration tasks because the directly provide gradients of the output variable by use of automatic differentiation.

We would like to emphasize that our methodology can in principle be applied to any model with finitely many parameters, from the classical Black Scholes or Heston models to the rough Bergomi model of [5], also to large class of rough volatility models (see Horvath, Jacquier and Muguruza [34] for a general setup). In fact the methodology is not limited to stochastic models, also parametric models of implied volatility could be used for generating training samples of abstract models, but we have not pursued this direction further. For the sake of concreteness, we give an example of one rough volatility model, since computational costs of available numerical methods are especially limiting for this model class.

Example 1. In the abstract model framework, the rough Bergomi model [5] is represented by $\mathcal{M}^{\mathrm{rBergomi}}(\Theta^{\mathrm{rBergomi}})$, with parameters $\theta = (\xi_0, \eta, \rho, H) \in \Theta^{\mathrm{rBergomi}}$. For instance, we may choose

$$\Theta^{\mathrm{rBergomi}} = \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times [-1, 1] \times]0, 1/2[,$$

to stay in a truly rough setting. The model corresponds to the following system for the log price X and the instantaneous variance V:

$$dX_t = -\frac{1}{2} V_t dt + \sqrt{V_t} dW_t, \quad \text{for } t > 0, \quad X_0 = 0, \tag{3a}$$

$$V_t = \xi_0(t) \mathcal{E}\left(\sqrt{2H}\eta \int_0^t (t-s)^{H-1/2} dZ_s\right), \quad \text{for } t > 0, \quad V_0 = v_0 > 0, \tag{3b}$$

where H denotes the Hurst parameter, $\eta > 0$, $\mathcal{E}(\cdot)$ the Wick exponential, and $\xi_0(\cdot) > 0$ denotes the initial forward variance curve (see [9, Section 6]), and W and Z are correlated standard Brownian motions with correlation parameter $\rho \in [-1, 1]$.

3 Deep calibration

In the following sections we elaborate on the objectives and advantages of this two step calibration approach and present examples of neural network architectures, precise numerical recipes and training procedures to apply the two step calibration approach to a family of stochastic volatility models. We also present some numerical experiments and report the learning errors compared to chosen parameters of the synthetic data.

There are several advantages of separating the tasks of pricing and calibration. Above all, the most appealing reason is that it allows us to build upon the knowledge we have gained about the models in the past decades, which is of crucial importance from a risk management perspective. By its very design, **(i)** deep learning the *price approximation* combined with **(ii)** deterministic calibration does not cause more headache to risk managers and regulators than the corresponding stochastic models do. Designing the training as described above demonstrates how deep learning techniques can successfully extend the toolbox of financial engineering, without imposing the need for substantial changes in our risk management libraries.

3.1 One-step approach: Deep calibration by the inverse map

A more and more popular approach in quantitative finance (and many other fields of engineering) is to develop purely data-driven frameworks, without relying on formal models. This approach leaves the meaning of calibrated network parameters unexplained, not to mention the ambiguity about the choice of the number of network parameters and network design. This can cause major challenges towards today's regulatory requirements. In addition, issues of generalizability – how can one price exotic options in a network trained with vanilla option data, to give a simple example – are difficult to analyse, and traditional paradigms of finance – such as no arbitrage – are hard to guarantee in the absence of a model.

A second, more model based approach was proposed in the pioneering work of Hernandez [30], followed by several other authors such as Stone [53], Dimitroff, Röder and Fries [14] and many others. A main characteristic of the neural network proposed by [30] is that option price approximation and parameter calibration are done in one step within the same network. Indeed, the idea is to directly learn the whole calibration problem, i.e., to learn the model parameters as a function of the market prices (typically parametrized as implied volatilities). In the formulation of Section 2, this means that we learn the mapping

$$\Pi^{-1} : (\mathcal{P}(\zeta))_{\zeta \in Z'} \mapsto \widehat{\theta}.$$

More precisely, [30] trains a deep neural network based on labelled data (x_i, y_i), $i = 1, \dots, N$, with

$$x_i = (\mathcal{P}(\zeta))_{\zeta \in Z'_i}$$

for day t_i (in the past) and the corresponding labels

$$y_i = \widehat{\theta}_i,$$

obtained from calibrating the model to the market data y_i using traditional calibration routines. The number of labelled data points N is, of course, limited to the amount of (reliable) historical market price data available.

In spite of the promising results by Hernandez [30] the main drawback of this approach, as Hernandez observes, is the lack of control on the function Π^{-1}. Furthermore, from a risk management perspective one has no guarantee how well the learned mapping of Π^{-1} will solve the calibration problem when exposed to unseen data. In fact, this is the behaviour observed in Hernandez [30], since the out of sample performance tends to differ from the in sample one, suggesting a not fully satisfactory generalisation of the learned map. We recover the same behaviour of the inverse map in our own experiments, which we included in Appendix A.

3.2 Two-step approach: Learning the implied volatility map of models

The two step approach is somewhere mid-way between a sole reliance on traditional pricing methods (Monte Carlo, finite elements, finite differences, Fourier methods, asymptotic methods etc.) and the direct approach described above that calibrate directly to the price data. Here, one separates the calibration procedure as described in Section 2 **(i)** We first learn (approximate) the pricing map by a neural network that maps parameters of a stochastic model to prices or implied volatilities. In other words, we set up and train (off-line) a neural network to learn the pricing map P. In a second step **(ii)** we calibrate (on-line) the model – as approximated by the neural network trained in step **(i)** – to market data using a standard calibration routine. To formalise the two step approach, for an option parametrized by ζ and a model $\mathcal{M}$ with parameters $\theta \in \Theta$ we write $\widetilde{P}(\theta, \zeta) \approx P(\theta, \zeta)$ for the approximation $\widetilde{P}$ of the true pricing map P based on a neural network. Then, in the second step, for a properly chosen distance function δ (and a properly chosen optimization algorithm) we calibrate the model by computing

$$\widehat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}}\, \delta \left(\left(\widetilde{P}(\theta, \zeta) \right)_{\zeta \in Z'}, \left(\mathcal{P}(\zeta) \right)_{\zeta \in Z'} \right). \tag{4}$$

In principle, this method is not unlike traditional calibration routines, as the true option price has to be numerically approximated for all but the most simple models. This particular approximation method tends to be orders of magnitudes faster compared to other numerical approximation methods for all tested models. In particular, note that the (slow) training stage of the neural network itself only has to be done once. We will come back to comparisons of actual computational times in the numerical section of this paper.

At this stage, we note that the deep calibration routine is not yet specified in any details: apart from purely numerical details such as the choice of the architecture of the neural networks, the loss functions and optimization algorithms of both the training of the neural networks in stage **(i)** and the actual calibration in stage **(ii)**, one particularly important choice is whether the neural network learns implied volatilities of individual options or rather a full implied volatility surface. Before discussing these details, let us already highlight some of the differences to the one-step approach of [30]. While the one-step approach is probably marginally faster, we see the main benefit of the two-step approach in the increased stability, which is influenced by two key differences:

- As the neural network is only responsible for option pricing in the model, synthetic data can (and should) be used for training. Hence, we can easily increase the number of training data, and the training data are completely unpolluted from market imperfections.
- The two-step approach induces a natural decomposition of the overall calibration error into a pricing error (from the neural network) and a model misfit to the market data. Hence, the performance of the neural network itself is generally independent of changing market regimes – which might, of course, change the suitability of the model under consideration.

These points, in particular, imply that frequent re-training of the neural network is not needed in the two-step approach.

3.2.1 The two step approach: Pointwise training and implicit and grid-based training

The underlying principle of the two-step approach appears in one way or another in a number of related contributions De Spiegeleer, Madan, Reyners and Schoutens [13] and McGhee [46]. In fact, the early works of Hutchinson, Lo and Poggio [38] and the more recent work of Culkin and Das [12]–where Deep Neural Networks are applied neural to learn the Black-Scholes formula–can be recognised as Step **(i)** of the two-step approach in a Black-Scholes context. Also Ferguson and Green [19] examine Step **(i)** of the two-step approach in [19] for basket options in a lognormal context and observe that the network even has a smoothing effect and increased accuracy in comparison to the underlying Monte Carlo prices. In this section, we examine its advantages and present an analysis of the objective function with the goal to enhance learning performance. Within this framework, the pointwise approach has the ability to asses the quality of $\widetilde{P}$ using Monte Carlo or PDE methods, and indeed it is superior training in terms of robustness.

Pointwise learning

Step (i): Learn the map $\widetilde{P}(\theta, T, k) = \widetilde{\sigma}^{\mathcal{M}(\theta)}(T, k)$ – that is in equation (4) above we have $\zeta = (T, k)$. In the case of vanilla options ($\zeta = (T, k)$) one can rephrase this learning objective as an implied volatility problem: In the implied volatility problem the more informative implied volatility map $\widetilde{\sigma}^{\mathcal{M}(\theta)}(T, k)$ is learned, rather than call- or put option prices $\widetilde{P}(\theta, T, k)$. We denote the artificial neural network by $F(w; \theta, \zeta)$ as a function of the weights w of the neural network, the model parameters θ and the option parameters ζ. The optimisation problem to solve is the following:

$$\widehat{\omega} := \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{N_{\text{Train}}} \eta_i (\widetilde{F}(w; \theta_i, T_i, k_i) - \widetilde{\sigma}^{\mathcal{M}}(\theta_i, T_i, k_i))^2. \tag{5}$$

where $\eta_i \in \mathbb{R}_{>0}$ is a weight vector.

Step (ii): Solve the classical model calibration problem

$$\widehat{\theta} := \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{m} \beta_j (\widetilde{\sigma}^{\mathcal{M}(\theta)}(\theta, T_j, k_j) - \sigma_{\text{BS}}^{\text{MKT}}(k_j, T_j))^2.$$

for some user specified weights $\beta_j \in \mathbb{R}_{>0}$, where now the (numerical approximation of the) option price $\widetilde{P}(\theta, T, k)$ resp. implied volatility $\widetilde{\sigma}^{\mathcal{M}(\theta)}(T, k)$ is replaced by the DNN approximation $\widetilde{F}(\widehat{\omega}; \theta, T, k)$ obtained in Step **(i)**.

The critical part is, of course, the first step, as the second one merely corresponds to classical calibration against liquid options. For the first step, key issues are the choice of training data and the architecture of the neural network. Regarding the training data, the general idea is as follows:

1. Choose realistic *"prior"* distributions for both model parameters θ and option parameters ζ ($= (T, k)$ in the above notation). The point is that many theoretically possible parameters are very unlikely to ever occur in real markets, for both model and option parameters. Hence, it is wasteful to spend resources to learn the pricing map for, say, maturities in the range of hundreds of years. The simplest choice is to simply impose uniform distributions on truncated

parameter ranges, but nothing prevents more "informed" possibilities, for instance taking into account historical distributions of estimated model parameter values or observed option parameter values.

2. Simulate model and option parameters according to the distribution chosen before and compute the corresponding option price or implied volatility, which serves as label for the respective parameter vector. The computation can be done for any available numerical method, for instance Monte Carlo simulation. As an aside, this mechanism can, of course, be used to produce training, testing and validation data in the sense of the machine learning literature.

Remark 3. Note that the above mentioned "informed" parameter distributions could also be encoded as weights into the loss function for the training of the neural network.

Remark 4. Instead of simulation of parameter values, we could also consider deterministic grids in the parameter space. In very high dimensional parameter spaces this probably becomes unfeasible due to the curse of dimensionality, but in the current context this approach may very well improve training of the neural network. We leave a comparison to future work.

Implicit & grid-based learning

We take this idea further and design an implicit form of the pricing map that is based on storing the implied volatility surface as an image given by a grid of "pixels". This image-based representation has a formative contribution in the performance of the network we present in Section 5. Let us denote by $\Delta := \{k_i, T_j\}_{i=1,\,j=1}^{n,\,m}$ a fixed grid of strikes and maturities, then we propose the following two step approach:

Step (i): Learn the map $\widetilde{F}(\theta) = \{\sigma_{BS}^{\mathcal{M}(\theta)}(T_i, k_j)\}_{i=1,\,j=1}^{n,\,m}$ via neural network where the input is a parameter combination $\theta \in \Theta$ of the stochastic model $\mathcal{M}(\theta)$ and the output is a $n \times m$ grid on the implied volatility surface $\{\sigma_{\text{BS}}^{\mathcal{M}(\theta)}(T_i, k_j)\}_{i=1,\,j=1}^{n,\,m}$ where $n, m \in \mathbb{N}$ are chosen appropriately (see Section 4.1) on a predefined fixed grid of maturities and strikes. $\widetilde{F}$ takes values in $\mathbb{R}^L$ where $L = \text{strikes} \times \text{maturities} = nm$. The optimisation problem in the image-based implicit learning approach is:

$$\widehat{\omega} := \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{N_{\text{Train}}^{\text{reduced}}} \sum_{j=1}^{L} \eta_j (\widetilde{F}(\theta_i)_j - \sigma^{\mathcal{M}}(\theta_i, T_j, k_j))^2, \tag{6}$$

where $N_{\text{Train}} = N_{\text{Train}}^{\text{reduced}} \times L$ and $\eta_i \in \mathbb{R}_{>0}$ is a weight vector.

Step (ii): Solve the minimisation problem

$$\widehat{\theta} := \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{L} \beta_j (\widetilde{F}(\theta)_i - \sigma_{\text{BS}}^{\text{MKT}}(T_i, k_i))^2,$$

for some user specified weights $\beta_j \in \mathbb{R}_{>0}$

The data generation stage for the image-based approach works as in the point-wise approach, except that the option parameters $\zeta = (T, k)$ are, fixed and are no longer part of the learning algorithms – except implicitly in the output/labels of the neural network. This is why they appear in the general objective function of pointwise learning (5) but no longer appear in the objective function (6) of

the grid-based learning above. In practice, we choose a grid Δ of size 8×11. By evaluating the implied volatility surface along 8×11 gridpoints with 40.000 different parameter combinations in Θ we effectively evaluate the "fit" of the surface as a whole. In our experiments we chose a 8×11 grid for practical reasons, but we are by no means limited to this number. For example, to obtain even higher accuracy, one could also choose a coarser grid, which would require longer learning time, but recall that learning only has to be done once. One advantage of a grid-based sampling is that one can re-use the same set of generated Monte Carlo paths along grid points. Once a grid is fixed one can also easily refine the grid by adding further refined points to it using the same set of Monte Carlo paths (evaluated at more time points).

Clearly, the neural network does depend on the grid Δ of option parameters ζ. Hence, we need to interpolate between gridpoints in order to be able to calibrate (in the calibration Step **(ii)**) also to such options, whose maturity and strike do not exactly lie on the grid Δ. While in the pointwise training the **interpolation between sampling points** is done by the network $\widetilde{F}(\theta)$ automatically (both in the model parameter space Θ and along the implied volatility surface in $\mathbb{K} \times \mathbb{T}$), in the grid-based implicit learning the network is only used for interpolation in the parameter space Θ, and it is implicit in the space dimension, that is, –based on smoothness assumptions of the implied volatility surface– we interpolate between gridpoints of the implied-volatility surface manually, using appropriate splines. This indirect dependence of the trained network on Δ is alluded to by the name "implicit learning".

Implicit smile-based learning:
–And outlook towards an implicit learning with more elaborate grids and tessalations of the IV surface–
We note that McGhee [46] follows an *implicit* approach for the lognormal SABR model, which lies somewhere between the pointwise and the image-based approaches of Step **(i)**: There, the inputs are $(\theta^{\text{SABR}}, T, k_1, \ldots, k_{10})$, and there are ten volatility outputs $\sigma_1, \ldots, \sigma_{10}$ per maturity T. Since between the reference points of the smile McGhee [46] also interpolates (by splines) based on a smoothness assumption of implied volatilities, we also refer to this approach as *implicit* training. The reference points $k_1, \ldots, k_{10}$ on the volatility surface are determined as a direct functional of the model parameters θ^{SABR} and of the maturity T, that is the learning is done slice-by slice. This sampling technique showcases an excellent working example of a *representative functional sampling* on the surface, where more samples are taken in certain regions of the surface, to ensure a good accuracy of the training in those regions (e.g. regions with higher liquidity). Though the sampling of the strikes in [46] is bespoke to the SABR model, it motivates the idea of *representative sampling grid (or tessalation net)*, which would be desirable to achieve also in a model agnostic context. We note that the introduction of the weight vectors $\eta_i \in \mathbb{R}_{>0}$ in the objective function (6) of the grid-wise approach has a similar effect as a higher sampling frequency of a neighbourhood/point.

3.2.2 The role of the objective function: Pointwise training versus implicit and grid-based training

Comparing the pointwise approach (characterised by the general objective function (5)) and the image-based approach (characterised by the objective function (6)), we find that both of them can be advantageous in certain situations. We highlight the connection between the two below, and elaborate on some of the respective advantages of each approach.

Equation (6) can be brought to the form of (5) equation by inserting (into (5)) the specification values $\theta = \theta'$, with

$$\theta'_1 = \theta_1, \ldots, \theta'_L = \theta_1, \theta'_{L+1} = \theta_2, \ldots,$$

and recalling that L = strikes $\times$ maturities and $N_{\text{Train}} = N_{\text{Train}}^{\text{reduced}} \times L$. Hence, the pointwise approach is more general than the image-based one.

With this in mind we make the general note, many of the various advantages and disadvantages of both approaches can, in principle, be mitigated by careful choice of the data generation mechanism (of the training and validation datasets) and the loss function in the training.

- The biggest difference, between pointwise and image based implicit learning procedures is that image based implicit learning requires an outside (implicit) interpolation between the learned implied volatilities in order to compute the implied volatility of an option with an arbitrary strike or maturity, not aligned with the grid. At face value, this is of course an advantage of the pointwise (explicit) approach, where the interpolation is rather performed by the deep neural network. On the other hand, we note that the function $(T, k) \mapsto \sigma^{\mathcal{M}}(\theta; T, k)$ (for fixed model parameters θ) is usually a very well understood smooth function. (At least for useful models, as the market implied vol surface arguably is nice and smooth.) This is not necessarily true for $\theta \mapsto \sigma^{\mathcal{M}}(\theta; T, k)$, which is not nearly as well understood for more modern sophisticated models such as rough Bergomi. Hence, we have much more confidence in applying standard interpolation in (T, k) rather than in θ, which also lives in a higher dimensional space. Hence, the outside interpolation may, in practice, not cause any difficulties.
- Indeed, this very same structure induces a reduction of variance in the training data for the image-based approach as compared to the pointwise approach. Formally speaking, in the image based approach only the model parameters are sampled, while the strike and maturities of the underlying instruments are deterministic. As a side note, keep in mind that we should always compare the two approaches based on a fixed number N_{Train} of total training data.
- It is also easier to take into account the structure of real financial data into the data generation for the pointwise approach by adjusting the (random) sampling distribution on the surface accordingly. Clearly, not all options are equally *important* for the purpose of calibration, but we would like to concentrate on liquid options. It is easy to adjust the sampling distribution for strikes and maturities in the pointwise approach to take into account historical numbers of liquidity. In the grid-based approach, this can to some extent be taken into account by the choice of the weight vector $\eta_i \in \mathbb{R}_{>0}$ in (6), or more accurately taken into account by using non-uniform, non-tensorized, or bespoke quasirandom sampling grids, with higher density of points in regions with higher liquidity.
- The image-based approach may be seen as an efficient dimension-reduction technique as compared to the pointwise one. Indeed, as dimensions are shifted from the input of the neural network to the output, the learning task becomes easier since lower-dimensional. Of course, the price we pay is that we only learn the values of the implied volatilities on a fix grid Δ of option parameters. In this example, this price is, however, worth paying since the regularity of the volatility surface is well understood. This implies that we know very well the number and location of grid points required to get good fits globally in terms of the chosen interpolation.

In the particular calibration example presented in Section 5 below, the image-based approach performed somewhat better than the pointwise approach, which indicates that the variance and dimension reduction features may be more important than the other aspects in the above comparison.

Remark 5. In principle, the two-step approach is also amenable to other numerical interpolation methods. For instance, we could also use Chebyshev interpolation to approximate model implied volatilities such as [25].

Remark 6. In line with Remark 5, we note that the image-based approach (in conjunction with the outside interpolation) is a hybrid between a pure DNN approximation such as the point-wise approach and a standard polynomial interpolation method, such as Chebyshev approximation, see [25] for example. Of course, other, more specialized interpolation methods on the implied volatility surface are also possible, for instance using the SVI volatility parameterization, see for example [39].

4 Practical implementation

We start by describing the approximation network (Step **(i)** of Section 3 with objective functions (5) and (6)) and leave the discussion of calibration (Step **(ii)**) for Section 4.2 below. While several realted works [38, 12, 46] have demonstrated that learning the pricing map (Step **(i)**) in the Black-Scholes model and in certain clasical stochastic volatility models (such as the lognormal SABR model in [46]) can be done to a satisfactory accuracy with a single hidden layer, the situation is–as often–more delicate in the case of rough volatility models. Since these models are highly nonlinear nature, they also require deeper networks for an accurate approximation of their pricing functional.

4.1 Network architecture and training

We present the architecture used for the grid-based approach in some detail, as this approach was used for most of the numerical examples below.

1. A fully connected feed forward neural network with 3 hidden layers and 30 nodes on each layers;
2. Input dimension = n, number of model parameters
3. Output dimension = 11 strikes$\times$ 8 maturities for this experiment, but this choice of grid can be enriched or modified.
4. The three inner layers have 30 nodes each, which adding the corresponding biases results on a number
$$(n+1)\times 30 + 3\times(1+30)\times 30 + (30+1)\times 88 = 30n + 5548$$
of network parameters to calibrate.
5. We choose the Elu $\sigma_{\text{Elu}} = \alpha(e^x - 1)$ activation function for the network.

We train the neural network using gradient descent, the so-called 'Adam' minibatch training scheme due to Kingman and Ba [41], which is a version of the Stochastic Gradient Descent algorithm. In the

following, w denotes the set of parameters – weights and biases – of a neural network $F = F(w, x)$. Given parameters $0 \leq \beta_1, \beta_2 < 1, \epsilon, \alpha$, initial iterates $u_0 := 0, v_0 := 0, w_0 \in \Omega$, the Adam scheme has the following iterates:

$$\begin{aligned} g_n &:= \nabla^w \sum_{i=1}^{m} \mathcal{L}\left(F(w_{n-1}, X_{n,m}^{\text{batch}}), F^*(X_{n,m}^{\text{batch}})\right) \\ u_{n+1} &:= \beta_1 u_n + (1-\beta_1) g_n \\ v_{n+1} &:= \beta_2 v_n + (1-\beta_2) g_n^2 \\ w_{n+1} &:= w_n - \alpha \frac{u_{n+1}}{1-\beta_1^{n+1}} \frac{1}{\sqrt{v_n/(1-\beta_2^{n+1})} + \epsilon}. \end{aligned}$$

4.2 The calibration step

Once the pricing map approximator $\widetilde{F}$ for the implied volatility is found, only the calibration step is left to solve. We use the Levenberg-Marquart algorithm as presented in Section 2.

4.2.1 Bayesian Analysis of the Calibration

Intuitively, we are interested in *quantifying the uncertainty* about model parameter estimates obtained by calibrating with the approximative implied volatility map map $\widetilde{F}$. To this end, we switch to a Bayesian viewpoint and treat model parameters θ as random variables. The fundamental idea behind Bayesian parameter inference is to update prior beliefs $p(\theta)$ with the likelihood $p(\boldsymbol{y} \mid \theta)$ of observing a given point cloud $\boldsymbol{y} \in \mathbb{R}^N$ of implied volatility data to deduce a posterior (joint) distribution $p(\theta \mid \boldsymbol{y})$ over model parameters θ.

Formally, for pairs (T_i, k_i) of time to maturity and log-moneyness, let an implied volatility point cloud to calibrate against be given by

$$\boldsymbol{y} = \left[y_1\left(T_1, k_1\right), \ldots, y_N\left(T_N, k_N\right)\right]^T \in \mathbb{R}^N$$

and analogously, collect model implied volatilities for model parameters θ

$$\widetilde{\boldsymbol{F}}\left(\theta\right) = \left[\widetilde{F}\left(\theta, T_1, k_1\right), \ldots, \widetilde{F}\left(\theta, T_N, k_N\right)\right]^T \in \mathbb{R}^N.$$

We perform a liquidity-weighted nonlinear Bayes regression. Mathematically, for heteroskedastic sample errors $\sigma_i > 0, i = 1, \ldots, N$, we postulate

$$\boldsymbol{y} = \widetilde{\boldsymbol{F}}\left(\theta\right) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}\left(0, \operatorname{diag}(\sigma_1^2, \ldots, \sigma_N^2)\right),$$

so that for some diagonal weight matrix $\boldsymbol{W} = \operatorname{diag}(w_1, \ldots, w_N) \in \mathbb{R}^{N \times N}$, the liquidity-weighted residuals are distributed as follows

$$\boldsymbol{W}^{\frac{1}{2}}\left[\boldsymbol{y} - \widetilde{\boldsymbol{F}}\left(\theta\right)\right] \sim \mathcal{N}\left(0, \operatorname{diag}(w_1\sigma_1^2, \ldots, w_N\sigma_N^2)\right).$$

In other words, we assume that the joint likelihood $p(\boldsymbol{y} \mid \theta)$ of observing data $\boldsymbol{y}$ is given by a multivariate normal. In absence of an analytical expression for the posterior (joint) probability $p(\theta|\boldsymbol{y}) \propto p(\boldsymbol{y}|\theta)p(\theta)$, we approximate it numerically using MCMC techniques [20] and plot the one- and two-dimensional projections of the four-dimensional posterior by means of an MCMC plotting library [21].

Remark 7. Of course, from a statistical point of view, loss functions of sum of squares form corresponds to a normality assumption on the error distribution when interpreted as an MLE, for instance. The normality assumption above, hence, merely mirrors the common choice of sum-of-squares as loss function for calibration in finance.

5 Numerical experiments

5.1 Speed and accuracy of the price approximation networks

As mentioned in Section 3.2 one crucial improvement win in comparison with direct neural network approaches, as pioneered by Hernandez [30], is the separation of (i) the implied volatility approximation function, mapping from parameters of the stochastic volatility model to the implied volatility surface–thereby bypassing the need for expensive Monte-Carlo simulations—and (ii) the calibration procedure, which (after this separation) becomes a simple deterministic optimisation problem.

Table 1 shows the CPU computation time for functional evaluation of a full surface under the rough Bergomi model of Example 1. Here, we take the forward variance ξ_0 as constant. In a future work we take a similar approach to constract a network that can consistently approximate a variaty of models including the the rough Bergomi model with a forward variance curve that is approximated (more generally) by piecewise constant function.

MC Pricing Full Surface	NN Pricing Full Surface	NN Gradient Full Surface	Speed up NN vs. MC
500.000 μs	$14,3$ μs	47 μs	$21.000 - 35.000$

Table 1: Computational time of pricing map (entire implied volatility surface) and gradients via Neural Network approximation and Monte Carlo (MC) for the image-based approach

Table 1 provides the speed of evaluating the trained neural network for the image-based approach, the numbers for the pointwise approach are very similar. We used

- Total number of parameteres: 5.668
- Training set of size 34.000 and testing set of size 6.000
- Rough Bergomi sample: $(\xi_0, \nu, \rho, H) \in \mathcal{U}[0.01, 0.16] \times \mathcal{U}[0.5, 4.0] \times \mathcal{U}[-0.95, -0.1] \times \mathcal{U}[0.025, 0.5]$
- Strikes: $\{0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5\}$
- Maturities: $\{0.1, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.0\}$

- Training data samples of Input-Output pairs are computed using Algorithm 3.5 in Horvath, Jacquier and Muguruza [34] with 60.000 sample paths and the spot martingale condition i.e. $\mathbb{E}[S_t] = S_0,\ t \geq 0$ as control variate.

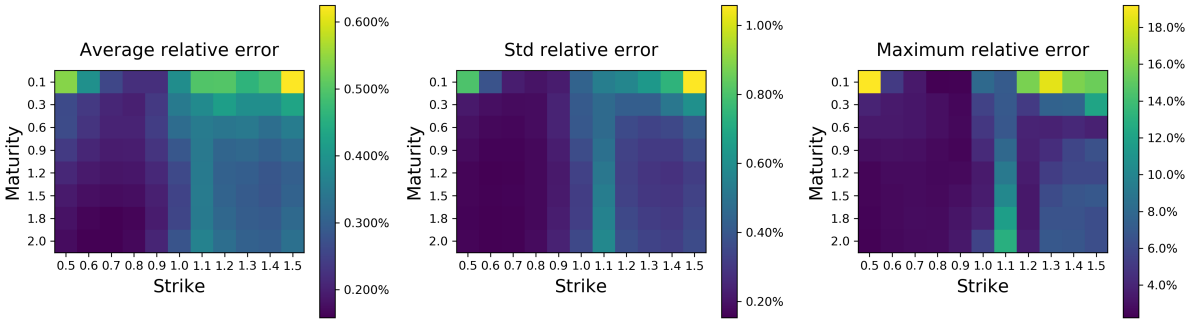


Figure 2: We compare surface relative errors of the neural network approximator against the Monte Carlo benchmark across all training data (34.000 random parameter combinations)in the rough Bergomi model. Relative errors are given in terms of Average-Standard Deviation-Maximum (Left-Middle-Right).

Figure 2 show that the average (across all parameter combinations) relative error[2] between neural network and Monte Carlo approximations is far less than 0.5% consistently (left image in Figure 2) with a standard deviation of less than 1% (middle image in Figure 2). Nevertheless, the maximum relative error goes as far as 25%. As previously stated, the beauty of this approach is the ability to asses whether the approximation is suitable and if not, where exactly fails or is more delicate. In this case, we observe that the approximation is less precise for short maturities and deep out-of-the-money/in-the-money options. Theses errors are consistent with the errors of the Monte Carlo training set .

5.2 Calibration speed and accuracy

To demonstrate the advantage of our two-step approach we obtain calibration times less than 40 milliseconds for the full implied volatility surface in the rough Bergomi model, which was notoriously slow to calibrate (several seconds) by Monte Carlo methods due to its non-Markovian nature. Note that these calibration times become much lower (usually under 10 milliseconds) for Markovian stochastic volaility models. This considerable speedup is due to the 21000-35000 factor speedup (reported in Tabe 1) of the approximation network.

In order to asses calibration the accuracy compared to synthetic data in a controlled experiment, the accuarcy of calibrated model parameters $\widehat{\theta}$ compared to the synthetically generated data with the set of parameters $\overline{\theta}$ that was chosen for the generation of our synthetic data. We measure the accuracy of the calibration via parameter relative error i.e.

$$E_R(\widehat{\theta}) = \frac{|\widehat{\theta} - \overline{\theta}|}{|\overline{\theta}|}$$

[2]Relative here is computed here as $|\sigma^{NN}(T,k) - \sigma^{MC}(T,k)|/|\sigma^{MC}(T,k)|$.

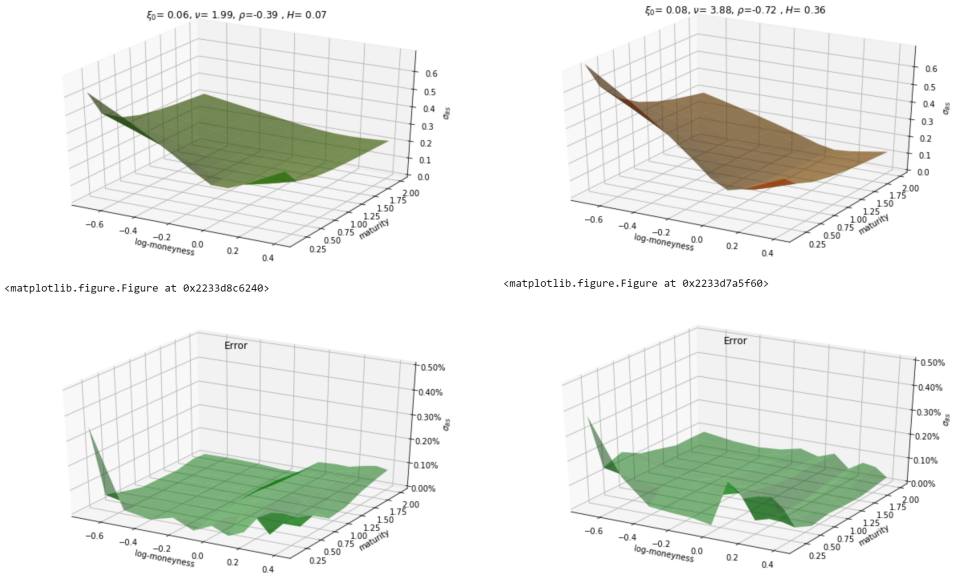


Figure 3: The Figure illustrates the distribution of the approximation error in space after the interpolation to a full implied volatility surface in two examples of model parameter choice.

as well as the root mean square error (RMSE) with respect to the original surface i.e.

$$\text{RMSE}(\widehat{\theta}) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(\widetilde{F}(\widehat{\theta})_{ij} - \sigma_{BS}^{MKT}(T_i, k_j))^2}.$$

Therefore, on one hand a measure of good calibration is a small RMSE. On the other hand, a measure of parameter sensitivity on a given model is the combined result of RMSE and parameter relative error. For this set of tests, we again restrict ourselves to the image-based approach for learning the price (implied volatility) function in the model.

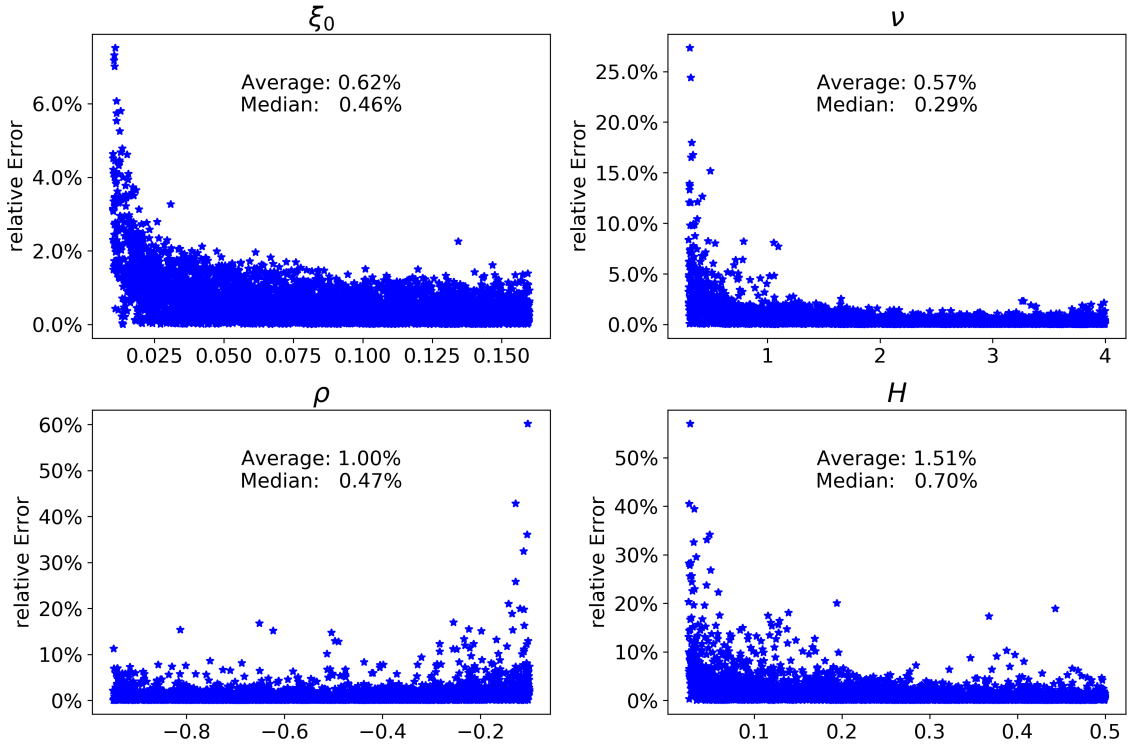


Figure 4: Calibration relative error per parameter in the test set in the rough Bergomi model

Figures 4 shows relative errors after calibration via Levengerg-Marquardt in the rough Bergomi model. We observe that largest errors are concentrated for small H or small vol of vol ν situations. Naturally, the relative error is more sensitive around 0 as well. Once again, we emphasise that by understanding the error zones of the pricing function P (see Figure 2) along with parameter relative errors in Figure 4, we are able to asses its quality and detect parameter configurations that might yield a lower performance of the calibration process.

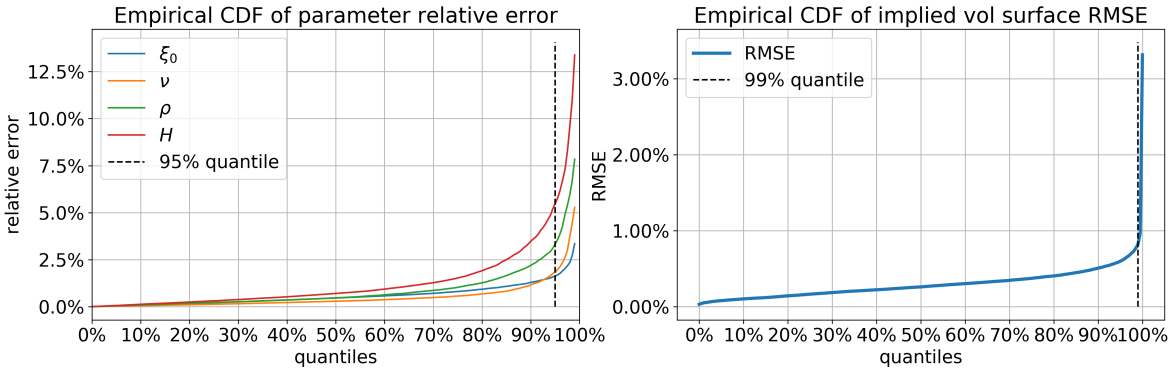


Figure 5: Cumulative Distribution Function (CDF) of Rough Bergomi parameter relative errors (left) and RMSE (right) after Levengerg-Marquardt calibration across test set random parameter combinations.

To finalise our analysis, Figure 5 shows that the 99% quantile of the RMSE is below 1%, even though parameter relative errors might be higher (see 4 as well), particularly when the parameters are close to 0. Notably, the maximum RMSE across the full surface (i.e. the 88 grid points) is below 4%, which suggests a surprisingly good accuracy.

5.3 A Bayes calibration experiment

We next test the deep calibration procedure using the Bayesian point of view sketched in Section 4.2.1. Here, we use the pointwise approach for learning the model implied volatility map. We perform two experiments. First, fixing $\theta = \theta^\dagger$, we generate a synthetic implied volatility point cloud

$$\boldsymbol{y}_{\text{synth}} = \left[P\left(\theta^\dagger, T_1, k_1\right), \ldots, P\left(\theta^\dagger, T_N, k_N\right)\right] \in \mathbb{R}^N$$

using Monte Carlo simulation as in Section 5.2 above. Next, we perform a non-weighted Bayesian calibration against the synthetic surface and collect the numerical results in Figure 6.

More precisely, the figure shows histograms from the posterior distribution of the one-dimensional marginal distribution of the (four-dimensional) parameter θ in the rough Bergomi model, together with contour plots of all pairs of two-dimensional marginal distributions based on kernel density estimates of the joint densities. The titles of the histogram-windows report the empirical medians together with the differences to the 2.5% and 97.5% quantiles, respectively. The dashed lines in the histogram plots show those quantiles.

If the map $\widetilde{F}$ is sufficiently accurate for calibration, the computed posterior should attribute a large probability mass around $\theta^\dagger$. The results in Figure 6 are quite striking in several ways: (1) From the univariate histograms on the diagonal it is clear that the calibration routine has identified sensible model parameter regions covering the true values. (2) Histograms are unimodal and its peaks close or identical to the true parameters. (3) The isocontours of the 2d Gaussian KDE in the off-diagonal pair plots for (η, H) and (η, ρ) show exactly the behaviour expected from the reasoning in the last

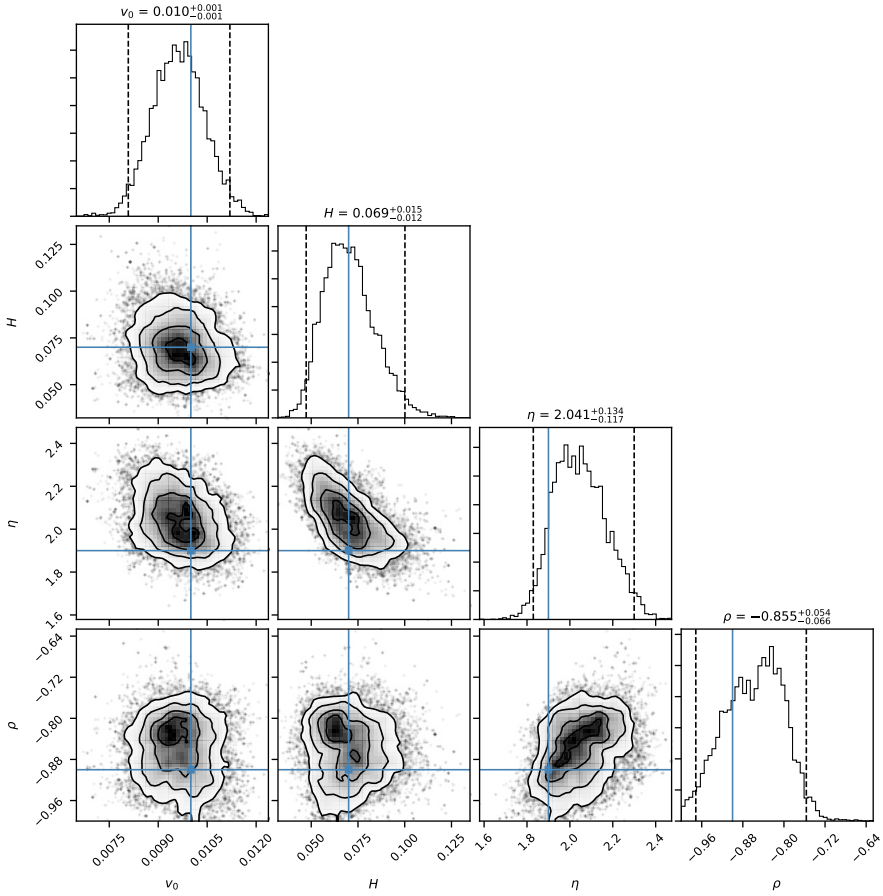


Figure 6: Bayes calibration against synthetic implied volatility surface computed for model parameters $\theta^\dagger$. Solid vertical blue lines indicate true parameter values.

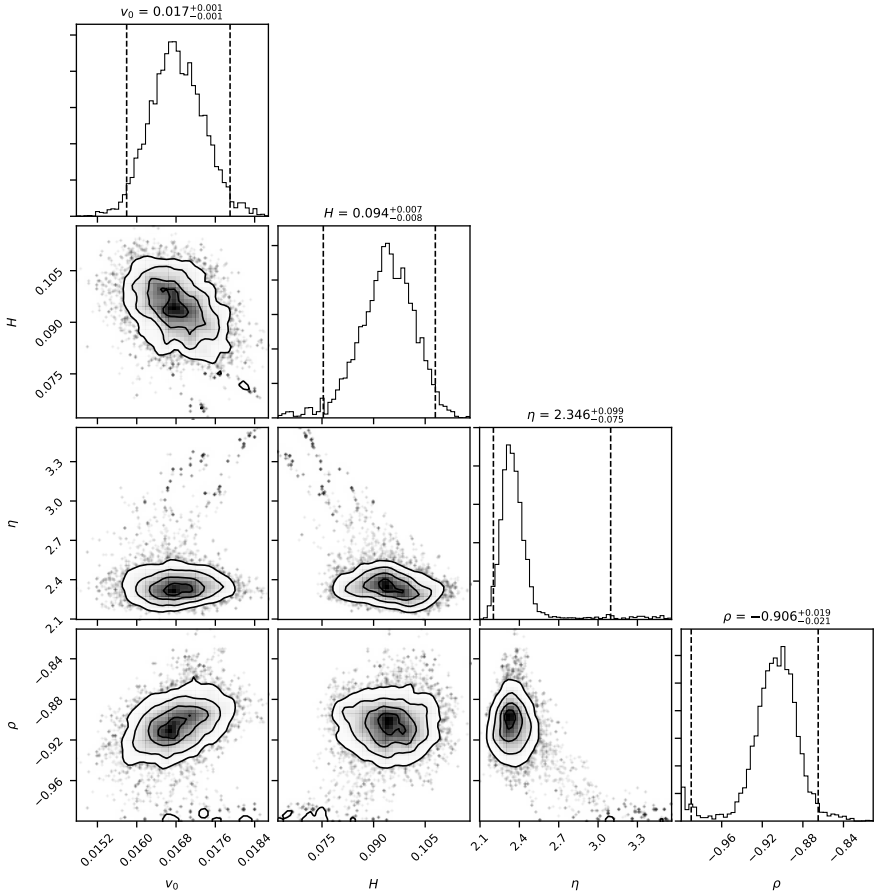


Figure 7: Liquidity-weighted Bayes calibration against SPX market implied volatility surface from 19th May 2017. Liquidity proxies given by inverse bid-ask-spreads.

section: Since increases or decreases in one of η, H or ρ can be offset by adequate changes in the others with no impact on the calculated IV, the Bayes posterior cannot discriminate between such parameter configurations and places equal probability on both combinations. This can be seen by the diagonal elliptic probability level sets.

In a second experiment, we want to check whether the inaccuracy of $\widetilde{F}$ allows for a successful calibration against market data. To this end, we perform a liquidity-weighted Bayesian regression against SPX implied volatilities from 19th May 2017. For bid and ask IVs $a_i > 0$ and $b_i > 0$ respectively, we proxy the IV of the mid price by $m_i := \frac{a_i+b_i}{2}$. With spread defined by $s_i = a_i - b_i \geq 0$, all options with $s_i/m_i \geq 5\%$ are removed because of too little liquidity. Weights are chosen to be $w_i = \frac{m_i}{a_i - m_i} \geq 0$, effectively taking inverse bid-ask spreads as a proxy for liquidity. Finally, σ_i are proxied by a fractional of the spread s_i. The numerical results in Figure 7 further confirm the accuracy of $\widetilde{F}$: (1) As can be seen on the univariate histograms on the diagonal, the Bayes calibration has again identified sensible model parameter regions in line with what is to expected. (2) Said histograms are again unimodal with peaks at or close to values previously reported in the literature. (3) Quite strikingly, at a first glance, the effect of the diagonal probability level sets in the off-diagonal plots as documented in Figure 6 cannot be confirmed here. However, the scatter plots in the diagrams do reveal some remnants of that phenomenon.

A A numerical experiment with the inverse map

To motivate the main drawbacks of the inverse map approach of Section 3.1, we calibrate rough Bergomi model with it, i.e., we consider the simple map

$$\Pi^{-1}(\Sigma_{\text{BS}}^{\text{rBergomi}}) \to (\hat{\xi_0}, \hat{\nu}, \hat{\rho}, \hat{H})$$

where $\Sigma_{\text{BS}}^{\text{rBergomi}} \in \mathbb{R}^{n \times m}$ is a rBergomi implied volatility surface and $(\hat{\xi_0}, \hat{\nu}, \hat{\rho}, \hat{H})$ the optimal solution to the corresponding calibration problem.
Remark 8. For simplicity we consider the strikes and maturities to be fixed for all implied volatility surfaces.

Inverse Map Architecture

- 1 convolutional layer with 16 filters and 3×3 sliding window
- MaxPooling layer with 2×2 sliding window
- 50 Neuron Feedforward Layer with *Elu* activation function
- Output layer with *linear* activation function
- Total number of parameters: 10.014
- Train Set: 34.000 and Test Set: 6.000
- $(\xi_0, \nu, \rho, H) \in \mathcal{U}[0.01, 0.16] \times \mathcal{U}[0.3, 4.0] \times \mathcal{U}[-0.95, -0.1] \times \mathcal{U}[0.025, 0.5]$
- strikes=$\{0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5\}$
- maturities=$\{0.1, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.0\}$

- Implied volatilities computed using Algorithm 3.5 in Horvath, Jacquier and Muguruza [34] with 60.000 sample paths and the spot martingale condition i.e. $\mathbb{E}[S_t] = S_0,\ t \geq 0$ as control variate.

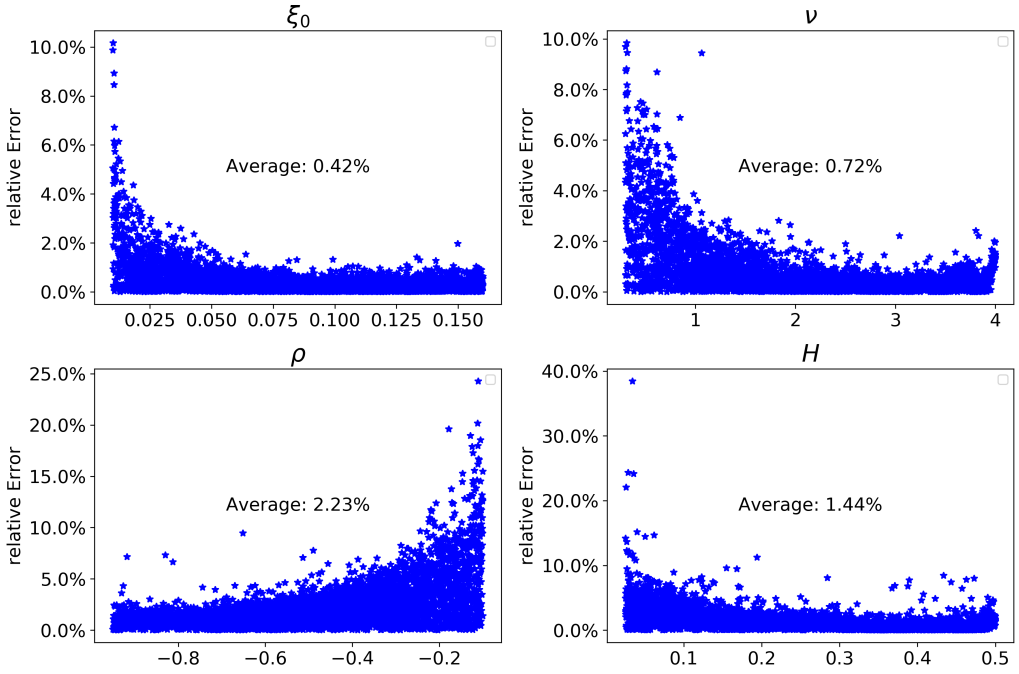


Figure 8: Out of sample relative errors per parameter calibration

Figure 8 shows that, indeed it is possible to approximate the inverse map and very sharply calibrate model parameters with a relatively small network. Convolutional networks make sense in this context, since a implied volatility surface has many features both in the strike and maturity direction that can be extracted, similar to image recognition problems. Notice also that the biggest error come from parameter configurations where the Monte Carlo input is more delicate i.e. very small H or very small volatility. Hence, the shape of the errors is intuitively natural and expected beforehand.

Black-Box function and "real" out of sample performance

Let us now consider "real" out of sample data, in the sense that it has not been generated by the rough Bergomi model itself. We generate implied volatility surfaces using the 2 factor Bergomi given by

$$dX_t = -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t$$
$$V_t = \xi_0(t)\mathcal{E}\left(\nu\left((1-\theta)\int_0^t \exp(-\kappa_X(t-s))dZ_s + \theta\int_0^t \exp(-\kappa_Y(t-s))dY_s\right)\right)$$

where W, Y and Z are correlated standard Brownian motions. We feed smiles from this model into our neural network to obtain the corresponding optimal rough Bergomi parameters. We then compare these values with a direct Monte Carlo calibration via Levenberg-Marquardt [43, 44] algorithm. Figure 9 shows that the neural network does not generalize properly out of sample and concludes that the brute force MC method clearly beats the Inverse map approach. The results by Hernandez [30] also support this conclusion, since his out of sample performance (based on different historical period) is reasonably worse than the in-sample one. However, we must emphasize that when the neural network is exposed to familiar situations i.e. surfaces close to the ones generated by the rBergomi model it may work better than the MC approach (see points below the dashed red line in Figure 9). This is likely due to delicate parameter configurations i.e. very low variance, where MC suffers to obtain accurate estimates whereas the network does not struggle that much.

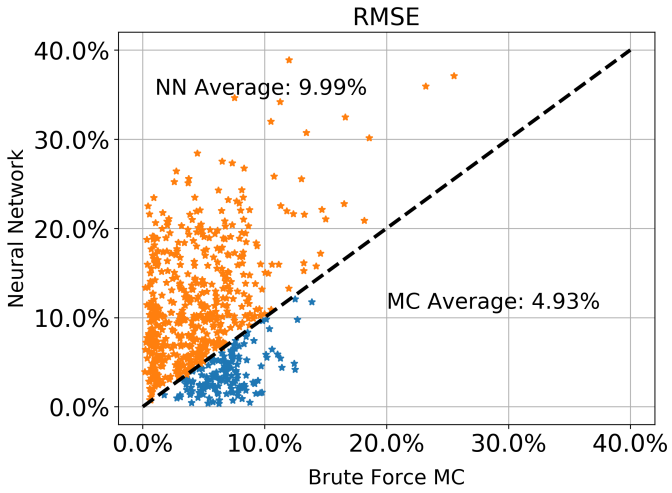
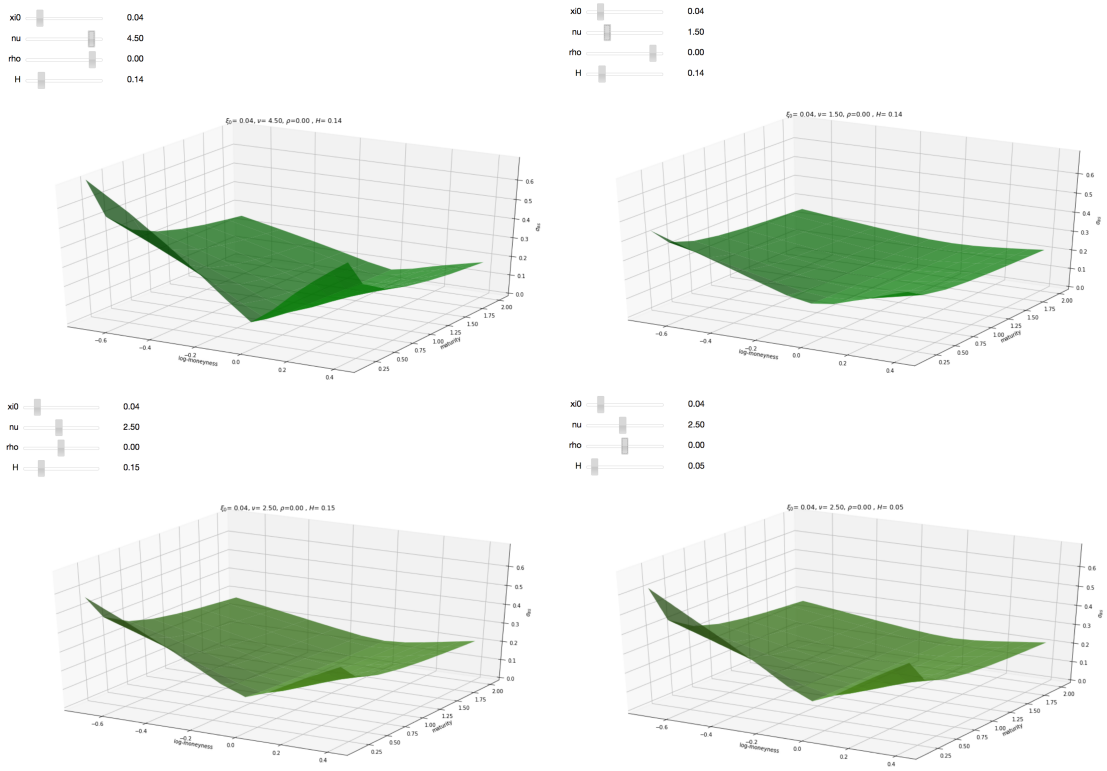


Figure 9: Stars represent the out of sample RMSE via neural network (NN) and brute force Monte Carlo (MC). Dashed black line represents the identity function.

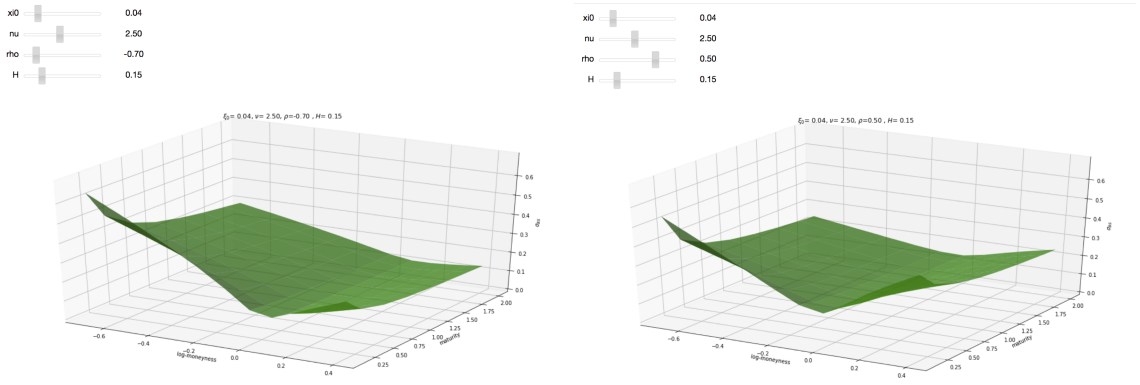
The *one-step* approach does not generalise the problem to all possible settings, since by design is not possible to train Π^{-1} on all possible (arbitrage-free) market scenarios. Moreover, there is a lack of understanding in the highly non-trivial function Π^{-1}, hence from a risk-managing perspective is more difficult to justify the use of this inverse approach than of the direct approach.



B Illusration of model parameters & the pricing engine in the rBergomi model

We showcase here the influence of the model parameters in the rough Bergomi model on the shape of the implied volatility surface using the hands-on pricing engine we generated via the DNN of **step (i)** for the rough Bergomi model. Our findings are in line with asymptotic results presented in [6] and with [47] for the role of the model parameters.

The model parameters ν, ρ and H correspond to the *smile* (ν), *skew* (ρ) and the *explosion (H) parameters* of the surface, while ξ_0 is the one-point approximation of the forward variance.
The images illustrate that the parameters ν and H influence the slope of the smile, and an explosive behaviour for short maturities can be achieved (without calibrating slice by slice) with a single surface if the Hurst parameter is $H << \frac{1}{2}$. And finally, as usual in stochastic volatility models, the parameter ρ introduces skewness in the surface as illustrated below.



References

[1] E. Alòs, J. León and J. Vives. On the short-time behavior of the implied volatility for jump-diffusion models with stochastic volatility. *Finance and Stochastics*, 11(4), 571-589, 2007.

[2] A. Antonov, M. Konikov, M. Spector. Modern SABR Analytics: Formulas and Insights for Quants, Former Physicists and Mathematicians. *SpringerBriefs in Quantitative Finance*, 2019.

[3] M. Avellaneda, A. Carelli, F. Stella. Following the Bayes path to option pricing. *Journal of Computational Intelligence in Finance*, 84, 1999.

[4] C. Bayer, P. Friz, P. Gassiat, J. Martin and B. Stemper. A regularity structure for rough volatility. arXiv:1710.07481, 2017.

[5] C. Bayer, P. Friz and J. Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6): 1-18, 2016.

[6] C. Bayer, P. Friz, A. Gulisashvili, B. Horvath and B. Stemper. Short-time near the money skew in rough fractional stochastic volatility models. arXiv:1703.05132, 2017.

[7] C. Bayer and B. Stemper. Deep calibration of rough stochastic volatility models. *Preprint*, arXiv:1810.03399

[8] M. Bennedsen, A. Lunde and M.S. Pakkanen. Hybrid scheme for Brownian semistationary processes. *Finance and Stochastics*, 21(4): 931-965, 2017.

[9] L. Bergomi. Stochastic Volatility Modeling. Chapman & Hall/CRC financial mathematical series. *Chapman & Hall/CRC*, 2015.

[10] B. Chen, C. W. Oosterlee and H. Van Der Weide. Efficient unbiased simulation scheme for the SABR stochastic volatility model, 2011.

[11] R. Cont. Model Calibration. *Encyclopedia of Quantitative Finance*, 5, 2010. DOI:10.1002/9780470061602.eqf08002

[12] R. Culkin and S. R. Das Machine Learning in Finance: The Case of Deep Learning for Option Pricing. *Journal of Investment Management*, github:BlackScholesNN 2017.

[13] J. De Spiegeleer, D. Madan, S. Reyners and W. Schoutens. Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting. SSRN:3191050, 2018.

[14] G. Dimitroff, D. Röder and C. P. Fries. Volatility model calibration with convolutional neural networks. *Preprint*, SSRN:3252432, 2018.

[15] R. Eldan and O. Shamir. The power of depth for feedforward neural neworks. *JMLR: Workshop and Conference Proceedings* `Vol 49:1-34`, 2016.

[16] O. El Euch and M. Rosenbaum. Perfect hedging in rough Heston models, to appear in *The Annals of Applied Probability*, 2018.

[17] M. Forde, H. Zhang. Asymptotics for Rough Stochastic Volatility

[18] J. Friedman, R. Tibshiran and T. Hastie. The Elements of Statistical Learning. *Springer New York Inc*, 2001.

[19] R. Ferguson and A. D. Green. Deeply learning derivatives. Preprint arXiv:1809.02233, 2018.

[20] D. Foreman-Mackey, D. W. Hogg, D. Lang, J. Goodman. emcee: the MCMC hammer, *Publications of the Astronomical Society of the Pacific*, 125(925), 306, 2013.

[21] D. Foreman-Mackey, corner.py: Scatterplot matrices in Python, *The Journal of Open Source Software* 24, `http://dx.doi.org/10.5281/zenodo.45906`, 2016.

[22] M. Fukasawa. Asymptotic analysis for stochastic volatility: martingale expansion. *Finance and Stochastics*, `15`: 635-654, 2011.

[23] J. Gatheral. The volatility surface: a practitioner's guide, *Wiley*, 2011.

[24] J. Gatheral, T. Jaisson and M. Rosenbaum. Volatility is rough. *Quantitative Finance*, `18`(6): 933-949, 2018.

[25] K. Glau, D. Kressner, and F. Statti. Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing. arXiv:1902.04367, 2019.

[26] A. Green. XVA: Credit, Funding and Capital Valuation Adjustments. *Wiley*, 2015.

[27] P. Hagan, D. Kumar, A. Lesniewski, and D. Woodward. Managing smile risk. *Wilmott Magazine*, `September issue: 84-108`, 2002.

[28] J. Han, A. Jentzen, E. Weinan. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning. *PNAS.*`115`(34) 8505-8510, August 2018.

[29] S.L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options, *The Review of Financial Studies* 6(2):327–343, 1993.

[30] A. Hernandez. Model calibration with neural networks. *Risk*, 2017.

[31] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, `2`(5):359-366, 1989.

[32] K. Hornik. M. Stinchcombe and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*. Vol. 3:11, 1990.

[33] P. Henry-Labordère. Deep primal-dual algorithm for BSDEs: applications of machine learning to CVA and IM. SSRN:3071506

[34] B. Horvath, A. Jacquier and A. Muguruza. Functional central limit theorems for rough volatility. arXiv:1711.03078, 2017.

[35] B. Horvath, A. Muguruza and T. Mehdi. Deep learning volatility. Available at SSRN 3322085, 2019.

[36] B. Horvath, O. Reichmann. Dirichlet Forms and Finite Element Methods for the SABR Model. *SIAM Journal on Financial Mathematics*, p. 716-754(2), May 2018.

[37] J. Hull and A. White. Pricing interest rate derivatives securities. *The Review of Financial Studies*)(3): 573-592, 1990

[38] J. M. Hutchinson, A. W. Lo and T. Poggio. A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. *The Journal of Finance*, 49(3) 851-889. *Papers and Proceedings Fifty-Fourth Annual Meeting of the American Finance Association, Boston, Massachusetts*, 1994.

[39] A. Itkin. To sigmoid-based functional description of the volatility smile. *Preprint*, arXiv:1407.0256, 2014.

[40] S. Ioffe and C. Szegedy. Batch normalisation: Accelerating deep network training by reducing internal covariate shift. *Preprint*, arXiv:1502.03167, 2015.

[41] D.P. Kingman and J. Ba, Adam: A Method for Stochastic Optimization. *Conference paper*, 3rd International Conference for Learning Representations, 2015.

[42] A. Leitao Rodriguez, L.A. Grzelak and C.W. Oosterlee. On an efficient multiple time step Monte Carlo simulation of the SABR model. *Quantitative Finance*, 17(10), pp.1549-1565, 2017.

[43] K. Levenberg. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics*. 2: pp. 164-168, 1944.

[44] D. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*. 11 (2): pp. 431-441,1963.‘

[45] S. Liu, A. Borovykh, L. A. Grzelak, C. W. Oosterlee. A neural network-based framework for financial model calibration. Preprint, arXiv:1904.10523, 2019.

[46] W. A. McGhee. An artificial neural network representation of the SABR stochastic volatility model. *Preprint*, SSRN:3288882, 2018.

[47] R. McCrickerd, M. Pakkanen, Turbocharging Monte Carlo pricing for the rough Bergomi model, *Quantitative Finance* 18(11):1877-1886, 2018.

[48] A. Leitao Rodriguez, A. Grzelak Lech, Cornelis W. Oosterlee. On a one time-step Monte Carlo simulation approach of the SABR model : Application to European options. *Applied Mathematics and Computation*, 293 p. 461-479, 2017,

[49] M. Sabate Vidales, D. Siska, L. Szpruch Unbiased deep solvers for parametric PDEs arXiv:1810.05094, 2018.

[50] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics* 375(15) 1339-1364, December 2018

[51] L. Setayeshgar, and H. Wang. Large deviations for a feed-forward network, *Advances in Applied Probability*, 43: 2, pp. 545-571, 2011.

[52] U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *Appl. Comput. Harmon. Anal.*, 44(3): 537-557, 2018.

[53] H. Stone. Calibrating rough volatility models: a convolutional neural network approach. *Preprint*, arXiv:1812.05315, 2018.