Deep Smoothing of the Implied Volatility Surface

Damien Ackerer
Swissquote Bank
Gland, Switzerland
damien.ackerer@swissquote.ch

Natasa Tagasovska
Department of Information Systems
HEC Lausanne
Switzerland
natasa.tagasovska@unil.ch

Thibault Vatter
Department of Statistics
Columbia University
New York, USA
thibault.vatter@columbia.edu

Abstract

We present an artificial neural network (ANN) approach to value financial derivatives. Atypically to standard ANN applications, practitioners equally use option pricing models to validate market prices and to infer unobserved prices. Importantly, models need to generate realistic arbitrage-free prices, meaning that no option portfolio can lead to risk-free profits. The absence of arbitrage opportunities is guaranteed by penalizing the loss using soft constraints on an extended grid of input values. ANNs can be pre-trained by first calibrating a standard option pricing model, and then training an ANN to a larger synthetic dataset generated from the calibrated model. The parameters transfer as well as the non-arbitrage constraints appear to be particularly useful when only sparse or erroneous data are available. We also explore how deeper ANNs improve over shallower ones, as well as other properties of the network architecture. We benchmark our method against standard option pricing models, such as Heston with and without jumps. We validate our method both on training sets, and testing sets, namely, highlighting both their capacity to reproduce observed prices and predict new ones.

1 Introduction

An *option* is a financial contract giving the option holder the right to buy (a call option) or the right to sell (a put option) an asset for a predetermined price (the *strike price*) on a predetermined date (the *expiry* date). Options can be written on various assets such as a stock, an index, a currency, a bond, or a commodity, and have been used since at least the Greek era for risk management and speculation. Nowadays, options are standardized and listed on exchanges where tens of millions of contracts are traded everyday.

An initial *premium* must be paid to the option seller in order to acquire today the right to buy or sell and asset in the future at, possibly, a preferential price. Intuitively, an option is more valuable when the underlying asset price is more likely to fluctuate, that is its volatility is higher. In seminal papers, Black and Scholes [7] and Merton [50] showed that by actively trading the underlying asset a market participant can replicate the option final payoff at expiry, perfectly and without risk. The so-called Black-Scholes (BS) formula thus allowed to determine the option premium, or price, and its discovery resulted in a burst in option trading activity. However, the formula builds on unrealistic assumptions such as continuous price trajectory and trading, absence of market frictions such as bid-ask spread and integer contract size, and normality of log-returns. Hence, practitioners and academics alike have

Preprint. Under review.

been passionate to develop new stock price models leading to more realistic option prices, however, in general they come at a higher computational cost and always with some theoretical limitations.

Practitioners commonly use the BS formula to price option because of its simplicity, yet they tune the *volatility parameter* so as to express their view on the stock return distribution. As a result, options with different strike prices and maturities may thus be associated with different volatility parameters. The *implied volatility surface (IVS)* is the continuous representation of this volatility parameter expressed as a function of the strike price and of the expiry. The IVS can be used to easily quantify the relative expensiveness of options with different characteristics, and to produce various measures of market-implied future returns. However, the IVS is observable only at a limited number of points. A major challenge when modeling the implied volatility surface is that the corresponding option prices should not allow arbitrage opportunities, that is constructing a portfolio that may generate profits at a zero initial cost. It is also worth noting that some commonly used models, such as SABR [24] and SVI [18], may not be arbitrage-free for some parameters values, see for example [53, Section 3], which contradicts real-life scenarios.

In this paper, we present a new methodology to smooth, interpolate, and extrapolate the implied volatility surface in an arbitrage-free way. We achieve this by modeling the implied variance with a multilayer artificial neural network (ANN) and penalizing the loss using soft constraints during training so as to prevent arbitrage opportunities. The soft-constraints specification is guided by theoretical results from Mathematical Finance. Smart initialization, or transfer, of the neural networks parameters can also be used when only sparse data is available. This can be achieved by first fitting a standard model to market prices and, second, training an ANN without constraints to reproduce the fitted model over an extended range of maturities and moneyness. Alternatively, one may directly use a previously trained ANN. Note that the implied volatility surfaces fluctuates almost continuously, which is one important distinction with standard ANN applications. Indeed, our problem is not high-dimensional but require frequent re-calibration.

We benchmark our method against standard models and study its performance both on training and testing sets, as well as on synthetic data, and real market prices of contracts on the S&P 500 index. Numerical experiments suggest that our method appropriately captures the features of standard option pricing models. We show that increasing model capacity generally leads to better fits, and that the soft constraints generally help decreasing the fitting error and the convergence speed. Similar results are obtained when applying our method to real data, where an ablation study shows that constrained learning helps producing better volatility surfaces, both in intrapolation and extrapolation.

The main ambition of this work is bridging the existing gap between a traditional challenge in finance and recent developments from the deep learning community, resulting in a trust-worthy, computationally efficient option pricing framework.

Short literature review. As neural networks and machine learning in general, prevail almost all aspects in science and industry, finance application have also been impacted [29, 20, 30, 9, 28, 27]. Deep learning have been studied with applications to option pricing in [54, 25, 17, 5, 39, 45, 46]. These papers exploit the well-known universal approximation property of neural networks [34, 33, 44]. Several applications of ANN models for implied volatility smoothing exist, see [14, 58, 47, 40, 57] with more details in Appendix A. Yet, these papers differ in that they either focus on modeling the price directly (attempting to hard-wire the absence of arbitrage opportunities directly in the ANN) or build upon a one-layer ANN with option price constraints to prevent arbitrage opportunities. However, hardwired constraints limit tremendously the ANN flexibility, and option price soft constraints require computationally costly and highly nonlinear transformations at each iteration. Instead, we rely on soft constraints involving the first and second derivatives of our ANN models which ease the computational burden. This work is the first to propose an arbitrage-free multilayer ANN construction of the implied volatility surface with easy-to-run non-arbitrage soft constraints.

Paper structure. The paper is organized as follows. Section 2 reviews the concept of implied volatility surface and discusses non-arbitrage conditions. We present our smoothing framework in Section 3. The numerical experiments and the empirical analysis can be found in Section 4 and Section 5 respectively. Section 6 concludes. More information on standard option pricing models, on implied volatility models, and on the experiments can be found in the Appendix.

2 The implied volatility surface

2.1 Background in option pricing models

We briefly review the Black-Scholes (BS) option pricing formula which is the baseline model used to extract the implied volatility from market option prices. For more details we refer the reader to standard finance textbooks such as [36, Chapter 15].

In the BSM model, the dynamics of the stock price S_t under the risk-neutral measure is given by

$$dS_t = (r - \delta)S_t dt + \sigma S_t dW_t \tag{1}$$

for some constants $r \in \mathbb{R}$, $\delta \geq 0$, and $\sigma > 0$, and where W_t is a standard Brownian motion. Let V_t denotes the price of a derivative at time t, then it satisfies the following partial differential equation,

$$0 = \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta)S\frac{\partial V}{\partial S} - rV. \tag{2}$$

Consider the call option payoff $(x - K)^+$ with strike K and maturity T. Solving the PDE (2) with boundary condition $V_T = (S_T - K)^+$ with $S_t = S$ gives the following formula for the time-t call option price C,

$$C(S,\, \sigma,\, r,\, \delta,\, K,\, T - t) = S^{-\delta(T-t)}\Phi(d_+) - \mathrm{e}^{-r(T-t)}K\Phi(d_-), \tag{3}$$

where $d_\pm = (\log(S/K) + (r - \delta)(T - t))/(\sigma\sqrt{T - t}) \pm (1/2)\sigma\sqrt{T - t}$.

The dynamics of stock prices in the real world do not follow a geometric Brownian motion. Empirically validated stylized fact of stock log returns are, for examples, stochastic volatility and leverage effect which are not capture by (1). Despite its shortcomings, the BSM model remains extremely popular in practice for its simple pricing formula, and the modeling complexity is moved to the input volatility parameter σ. Hence, if one understands the model and its limitations, the BSM formula can be used as a Rosetta Stone to analyze market prices.

Let $\pi(K, \tau)$ denotes the market price of a call option with time to maturity $T - t = \tau > 0$ and strike price $K \geq 0$. The main objective of this paper is modeling the implied volatility whose definition is given below.

Definition 2.1 *The* implied volatility $\sigma_{\mathrm{IV}}(k, \tau) > 0$ *is given by the equation*

$$\pi(K, \tau) = C(S,\, \sigma_{\mathrm{IV}}(k, \tau),\, r,\, \delta,\, K,\, \tau), \quad \textit{with the log moneyness } k = \log(K/S). \tag{4}$$

The implied volatility surface *is given by* $\sigma_{\mathrm{IV}}(k, \tau)$ *for* $k \in \mathbb{R}$ *and* $\tau > 0$.

For fixed $\tau > 0$, then $\sigma_{\mathrm{IV}}(k, \tau)$ for $k \in \mathbb{R}$ defines a volatility smile. If the smile has a *U shape*, then the tail of the log return $\log(S_T/S_t)$ distribution are thicker than the tails of the Gaussian distribution, and vice versa. If the smile exhibits a skew, then one side of the log return distribution is thicker than the other. For example, if the left side of a smile, which is a slice of the surface for a fixed τ, is steeper than the right side, then the log price is more likely to experience large losses than large gains.

2.2 Aribtrage-free surface

A static arbitrage is a static trading strategy that has: a value that is both zero initially and always greater than or equal to zero afterwards, and a non-zero probability of having a strictly positive value in the future. In other words, an arbitrage costs nothing to implement while only providing upside potential, that is, it represents a risk-free investment after accounting for transaction costs. Under the assumption that economic agents are rational, any such opportunity should be instantaneously exploited until the market is arbitrage free. Therefore, option pricing models are designed in such a way that their call price surface $\pi(K, T)$ offers no possibility to implement such a strategy.

One can show that $\pi(K, T)$ is arbitrage-free if and only if it is free of calendar spread arbitrage and each time slice if free of butterfly arbitrage. A *calendar spread* is a strategy where one buys a call with a given maturity T_1 and sells another call with maturity T_2, both using the same strike, and where $T_1 > T_2$. At T_1, the value of the short call is $-\max(S_{T_1} - K, 0)$, whereas that of the long call, $\pi(K, T_2 - T_1)$, is always greater. A *butterfly* is a strategy where one buys two calls with

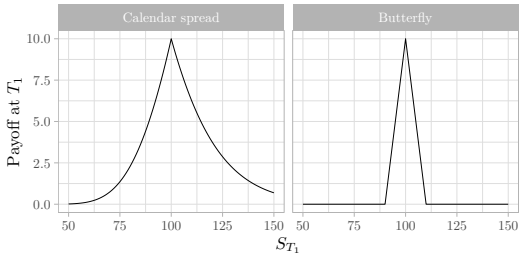

Figure 1: Payoffs of the calendar spread and butterfly as a function of the underlying asset price. For the calendar spread, $T_2 - T_1 = 1$, $K = 100$, and $\sigma = 0.25$. For the butterfly, $K_1 = 90$, $K_2 = 110$, and $K_3 = 100$.

strikes $K_1 < K_2$, and sells two other calls with strike $K_3 = (K_1 + K_2)/2$, but the same maturity. In Figure 1, we show the payoffs for each of the two strategies at T_1. Since the payoffs are always positives, they must have a nonzero initial price, for the market would otherwise allow for arbitrage opportunities.

The absence of arbitrage translates into constraints on the call price surface $\pi(K, T)$, which in turn can be expressed as conditions that the implied volatility surface $\sigma_{\text{IV}}(k, \tau)$ must satisfy [53, 19]. To write those conditions, we define the *total variance*,

$$\omega(k, \tau) = \sigma^2_{\text{IV}}(k, \tau)\, \tau. \tag{5}$$

Proposition 2.2 *Roper [53, Theorem 2.9] Let $S > 0$, $r = \delta = 0$, and $\omega : \mathbb{R} \times [0, \infty) \mapsto \mathbb{R}$. Let ω satisfy the following conditions:*

- *C1) (Positivity) for every $k \in \mathbb{R}$ and $\tau > 0$, $\omega(k, \tau) > 0$.*
- *C2) (Value at maturity) for every $k \in \mathbb{R}$, $\omega(k, 0) = 0$.*
- *C3) (Smoothness) for every $\tau > 0$, $\omega(\cdot, \tau)$ is twice differentiable.*
- *C4) (Monotonicity in τ) for every $k \in \mathbb{R}$, $\omega(k, \cdot)$ is non-decreasing, $\ell_{\text{cal}}(k, \tau) = \partial_\tau \omega(k, \tau) \geq 0$, where we have written ∂_τ for $\partial/\partial\tau$.*
- *C5) (Durrleman's Condition) for every $\tau > 0$ and $k \in \mathbb{R}$,*

$$\ell_{\text{but}}(k, \tau) = \left(1 - \frac{k\, \partial_k \omega(k, \tau)}{2\omega(k, \tau)}\right)^2 - \frac{\partial_k \omega(k, \tau)}{4}\left(\frac{1}{\omega(k, \tau)} + \frac{1}{4}\right) + \frac{\partial^2_{kk} \omega(k, \tau)}{2} \geq 0,$$

  where we have written ∂_k for $\partial/\partial k$ and ∂_{kk} for $\partial^2/(\partial k \partial k)$
- *C6) (Large moneyness behaviour) for every $\tau > 0$, $\lim_{k \to \infty} d_+(k, \omega(k, \tau)) = -\infty$.*

Then, the resulting call price surface is free of static arbitrage.

C1) and C2) are necessary conditions that any sensible model must satisfy. As for C3), it is merely sufficient to prove an absence of arbitrage when C4), C5), and C6) are also satisfied. Note that, assuming C3), C4) (respectively C5) and C6)) is satisfied if and only if the call price surface is free of calendar spread (butterfly) arbitrage [19].

3 Smoothing with neural networks

3.1 Architecture and learning algorithm

We describe here the base architecture and learning strategy that we follow to construct and train the neural networks, and will further refine them in Sections 3.2 and 3.3.

At a given time we observe N triplets $(\sigma_i,\, k_i,\, \tau_i)$, for $i = 1, \ldots, N$, where σ_i is the market implied volatility, k_i the log moneyness, and τ_i the time to maturity. In addition, we complement the sample with N_1 synthetic pairs $(k_i,\, \tau_i)$, for $i = N + 1, \ldots, N + N_1$. We denote the feature, or explanatory variable, $X_i = (k_i, \tau_i)$ and the target, or response variable, $Y_i = \sigma_i^2$ for $i = 1, \ldots, N + N_1$.

Let $F_\theta : \mathbb{R}^2 \mapsto \mathbb{R}$ be a standard feedforward multilayer neural network where θ is the set of network parameters to be trained. The mapping F_θ is thus defined by the multiple function composition

$$F_\theta(X) = \left(\bigcirc_{i=1}^{n+1} f_i^{W_i, b_i} \right)(X) \tag{6}$$

$$f_i^{W_i, b_i}(x) = \begin{cases} g_i(W_i\, x + b_i) & i = 1, \ldots, n \\ \exp(W_{n+1})\, x + \exp(b_{n+1}) & \text{otherwise} \end{cases} \tag{7}$$

where W_i is a weight matrix, b_i a bias vector, g_i an activation function applied element-wise, and n is the number of layers. We further impose that the last activation function takes non-negative values, that is $g_n : \mathbb{R} \mapsto \mathbb{R}_+$. Let n_i^{r} and n_i^{c} denote the number of rows and columns of W_i respectively. Note that the mapping in eqs. (6) and (7) is well defined if and only if $n_1^{\mathrm{c}} = 2$, $n_i^{\mathrm{r}} = n_{i+1}^{\mathrm{c}}$ for $i = 1, \ldots, n$, and $n_{n+1}^{\mathrm{r}} = 1$. Our predictor for the implied volatility is therefore given by $\hat{\sigma}_{\mathrm{IV}}(k, \tau) = \sqrt{F_\theta(X(k, \tau))}$. Note that, if g_n takes values in a bounded interval, then $\hat{\sigma}_{\mathrm{IV}}(k, \tau)$ is also bounded.

We fit the network parameters $\theta = \{W_1,\, b_1, \ldots, W_{n+1},\, b_{n+1}\}$ by minimizing the loss function

$$\mathcal{L}(\theta) = \mathcal{L}_0(\theta) + \sum_{j=1}^{m} \lambda_j\, \mathcal{L}_j(\theta) \tag{8}$$

where the term $\mathcal{L}_0(\theta)$ is a prediction error cost, the terms $\mathcal{L}_i(\theta)$ for $j = 1, \ldots, m$ materialize soft constraints aiming to ensure that the shape of $\{F_\theta(X);\, X \in \mathbb{R} \times \mathbb{R}_+\}$ is indeed a sensible implied volatility surface, and λ_i for $j = 1, \ldots, m$ are the corresponding penalty weights. We let the prediction error be the sum of the root-mean-squared-error (RMSE) and the mean-absolute-percentage-error (MAPE),

$$\mathcal{L}_0(\theta) = \left(\frac{1}{N} \sum_{i=1}^{N} (Y_i - F_\theta(X_i))^2 \right)^{1/2} + \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i - F_\theta(X_i)|}{Y_i},$$

so as to penalize both absolute and relative errors. This is because the target $Y = \sigma_{\mathrm{IV}}^2(k, \tau)$ may take values of different levels.

Remark 3.1 *An alternative approach to impose shape constraints on the mapping F_θ is to hard-wire them into the neural network architecture, as in [14] for example. However, hard constraints are difficult to impose on multilayer neural networks, may reduce the neural network's flexibility, and may lead to more challenging leaning routines, see [49].*

3.2 Non-arbitrage conditions

In what follows we explain how each of the constraints/conditions in Proposition 2.2 can be handled either by refining the architecture of the neural network, or by adding a penalty term to the loss function (8). Note that conditions **C1)–C2)** are satisfied by design of the ANN. The mapping F_θ is twice differentiable as long as the activation functions g_i are twice differentiable, in which case **C3)** is satisfied. In practice, however, one may want to use the ReLU activation functions in which case F_θ is piece-wise linear and non-differentiable at a finite number of points. We note that ReLUs are a common choice in modern NNs, despite not being differentiable when the input is exactly zero, [1]

Condition C4). To prevent calendar arbitrage, we add to the total loss the following soft constraint

$$\mathcal{L}_1(\theta) = \frac{1}{N_1} \sum_{i=N+1}^{N+N_1} \max\left(0,\, -\ell_{\mathrm{cal}}(k_i,\, \tau_i)\right). \tag{9}$$

Condition C5). To prevent butterfly arbitrage, we add to the total loss the following soft constraint

$$\mathcal{L}_2(\theta) = \frac{1}{N_1} \sum_{i=N+1}^{N+N_1} \max\left(0,\, -\ell_{\mathrm{but}}(k_i,\, \tau_i)\right). \tag{10}$$

[1] Software implementations return one of the derivatives either side of zero when the input corresponds to the undefined point rather than raising an error [21].

Note that with the ReLU activation function we have that $\partial_{kk} F_\theta(X) = 0$ everywhere at a finite number of points.

Condition C6). This asymptotic constraint is equivalent to having that $\sigma^2(k,\tau)/|k| \in [0,2]$ when $k \to \pm\infty$, see [43, 6]. Therefore, we work only with activation functions that are at most linear which already guarantees that $F_\theta(X)\tau/|k| < \infty$ when $k \to \pm\infty$ for any $\tau > 0$. We may also add to the total loss the following soft constraint

$$\mathcal{L}_3(\theta) = \frac{1}{N_k} \sum_{i=N+1}^{N+N_1} \max\left(0, \frac{F(X_i)}{|k_i|} - 2\right) \mathbb{1}_{\{|k_i|>C\}} \quad \text{with } N_k = \sum_{i=N+1}^{N+N_1} \mathbb{1}_{\{|k|_i>C\}} \tag{11}$$

for some constant $C > 0$. In practice we only care about bounded values for k and τ, therefore we do not take include C6) unless stated otherwise (that is we set $\lambda_3 = 0$).

We continue with implementation choices which significantly improve our proposed ANN-based smoothing procedure.

3.3 Transfer and regularization

Transfer. The ANN models may be difficult to initialize and a lot of time may be lost during the learning process to approach a reasonable implied volatility model. Hence if domain knowledge is available for the problem at hand, it can be leveraged to accelerate the learning process and make the model more robust to the speciffic setting. Similar ideas have been proposed in [37, 38]. Furthermore, one may have a favorite candidate model in mind. For these two reasons, we suggest to pre-train ANN models on simulated data in order to efficiently re-train models as new data arrives. The parameters transfer also precondition the extrapolation behavior of the ANN models, as shown in numerical examples.

Model regularization. It might be important to regularize the model so as to keep its prediction behavior as close as possible to a reference model. Therefore, we may add to the total loss the following term

$$\mathcal{L}_4(\theta) = \sum_{i=1}^{n+1} \delta_{b,i} \left\|\bar{b}_i - b_i\right\|_2 + \delta_{W,i} \left\|\bar{W}_i - W_i\right\|_2 \tag{12}$$

where $\|x\|_2$ is the Euclidean-norm of x, for some reference weights $\bar{W}_i$, $\bar{b}_i$, for $i = 1, \ldots, n+1$, and some binary functions $\delta_{\cdot,i} \in \{0,1\}$ selecting the weights to regularize. The standard L^2-regularization is retrieved by setting $\bar{W}_i = 0$, $\bar{b}_i = 0$ for $i = 1, \ldots, n+1$, and $\delta \equiv 1$. The possibility to regularize the weights more finely may be useful, as illustrated in the following example.

Example (Black-Scholes prior). Set $\bar{W}_i = 0$, $\bar{b}_i = 0$ for $i = 1, \ldots, n$, $\bar{W}_{n+1}$, and $\delta_{b,i} = 0$ and $\delta = 1$ otherwise. Then F_θ converges to a flat implied volatility model when $\lambda_4 \to \infty$.

4 Numerical experiments

Evaluation metric and experimental setup. The method of the previous section is implemented using tensorflow [1] and available in the supplementary material. The total loss $\mathcal{L}(\theta)$ is minimized with the Adam Optimization Algorithm [42], and the parameters initialized randomly with uniform random variables on the compact $[-1,1]$ with the exception of b_{n+1} which is initialized at $\log(20\%)$. As adaptive learning rate and early stopping have shown to significantly improve training [26, 23, 56, 12, 52, 32], we follow this approach. Starting with a learning rate of 0.01, we let it decrease by a factor 2 on plateaus of length 500 epochs when the total loss was not improved by more than 1%. The learning routine stops if $\mathcal{L}(\theta)$ has not improved by 1% over 2000 epochs, and restarts using the initial learning rate until 10000 total epochs have been reached or until the total loss (8) is below 1%.

Convergence. To explain the large number of epochs, note that the signal-to-noise ratio for empirical volatility surfaces is generally large, and overfitting is seldom the main concern in this context. Furthermore, because datasets contain at most a few thousand samples at most and only two features (moneyness and time-to-maturity), we are not using minibatch.

Feature engineering. We observed that the inclusion of features inversely proportional to the time to maturity helped to calibrate the ANN models with fewer layers and neurons. We conjecture that

Table 1: Settings for the penalties

Condition	λ_1	λ_2
1	1	1
2	10	10
3	0	0

this is because the implied volatility surface tends to sharply increase with $|k|$ at short horizons while being more flat at longer horizons. Therefore, we include an initial static layer f_0 defined by

$$X = f_0(k, \tau) = \left(k,\, \tau,\, k\tau^{-0.5},\, k\tau^{-0.95}\right). \tag{13}$$

Note that conditions C1)–C2) remain satisfied with these features, and that the total variance remains asymptotically linear in k. The input dimension is now $n_1^{\mathrm{c}} = 4$.

Calibration models and Baselines. To study the properties of our approach in a controlled setting, we create two synthetic datasets using the Heston and Bates models, see Appendix B.1. We use the following vector of moneyness values $[0.3, 0.4, 0.6, 0.8, 0.9, 0.95, 0.975, 1, 1.025, 1.05, 1.1, 1.2, 1.3, 1.5, 1.75, 2, 2.5, 3]$, and maturities of half a week, one, two and three weeks, one to twelve months, eighteen months and two years.

Parameters common to the Heston and Bates model are $V_0 = 0.10^2$, $\theta = 0.25^2$, $\rho = -0.75$, $\kappa = 0.5$, and $\sigma = 1$. Jump parameters specific to the Bates model are $\lambda = 0.1$, $\beta = -0.05$, and $\alpha = 0.15$. We also set the interest rate and dividend yield to zero.

Model flexibility check. We first show that, as expected, the ANN model can reproduce any surface given enough parameters. In Appendix C.1 we show that the ANN model with four layers, 20 neurons per layer, and without arbitrage constraints is flexible enough to fit synthetic data and market data. In the following we will therefore use this configuration as an upper bound in terms of flexibility, focusing on the impact of arbitrage constraints and architecture choice, i.e. number of layers and neurons.

Losses convergence. We study the loss and speed of convergence for different configurations. We calibrate models with two and four hidden layers, five and twenty neurons per layers, and using the ReLu activation function. In Table 1, we summarize the different conditions for the arbitrage penalties: λ_1 (respectively λ_2) is the weight corresponding to (9) (respectively (10)). The first condition represents a vanilla case, where the fit and arbitrage related losses have an equal weight. In the second condition, absence of arbitrage is enforced more strictly by increasing the weight of the penalties. In the third condition, we perform an ablation study to see how well we do when absence of arbitrage is not enforced.

Figure 2 displays the different loss terms in (8). First, we see that increasing the number of layers or of neurons per layers generally decreases the RMSE and MAPE. Second, we observe that, when the number of layers is smaller, increasing the neurons per layer without enforcing more strictly the constraints can generate arbitrage opportunities. But such opportunities disappear when absence of arbitrage is given a higher weight.

Figure 3 displays the total number of iteration necessary before the calibration algorithm stops. As expected, increasing the number of layers or of neurons per layers leads to slower convergence. Interestingly, enforcing absence of arbitrage more strictly decreases the number of iterations required to converge.

Results for the Heston model can be found in Appendix C.2.

Smoothing behaviors. A major challenge is the extrapolation of the implied volatility surface to unobserved data points. Flexible non-parametric methods typically fail as they lack the guidance of a parametric model. In Figure 4 we calibrate ANN models with different loss terms and parameters initialization on a subsample of 100 options with narrow moneyness on the synthetic Bates data. The three models fit the train set almost perfectly. Yet, we see that the ANN without arbitrage losses, $\lambda_1 = \lambda_2 = 0$, fails extrapolate in a nonsensical way. On the other hand the arbitrage-free model with $\lambda_1 = \lambda_2 = 10$ behaves very smoothly. The last model is initialized with the weights of an ANN model fitted to a Bates model itself fitted on the train set, see Appendix B.3. It is furthermore penalized so that its weights remain close to its initialization weights with $\lambda_3 = 1$. This confirms that

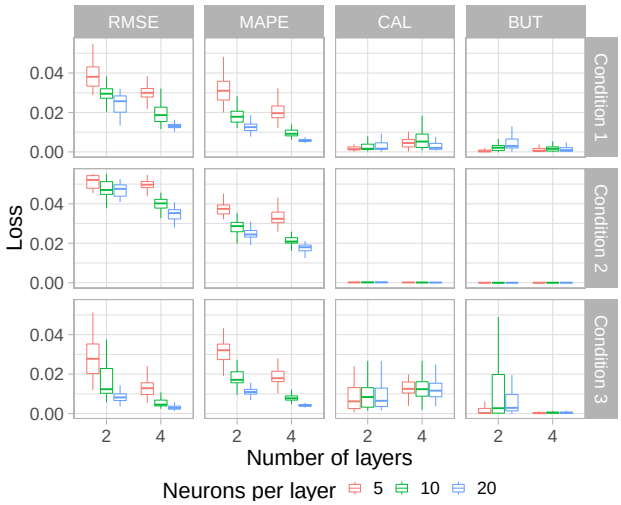

Figure 2: Losses for different combination of number of layers and neurons per layer. Synthetic data is generated from the Bates model.

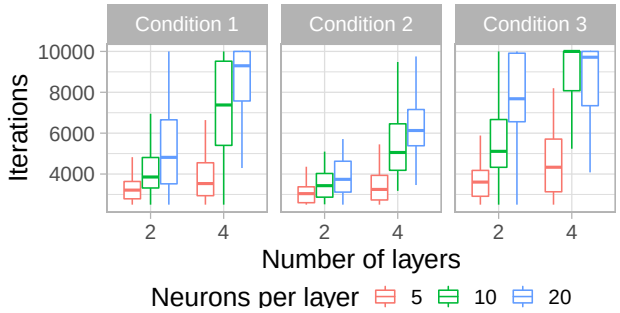

Figure 3: Number of iterations before the calibration stops for different combination of number of layers and neurons per layer. Synthetic data is generated from the Bates model.

a regularized learning can produce an extrapolation mimicking the Bates model, with most notable differences in this example for the positive log-moneynesses.

5 Empirical analysis

In this section, we apply our approach to modeling implied volatility surfaces extracted from S&P500 options prices. More specifically, we use a dataset containing the price of all options traded during January 2018, namely 61812 contracts, and Table 2 displays daily summary statistics.

We perform the following exercise for each day in the sample. First, we split the daily sample into a training and a testing set. Second, we fit the model on the training set and evaluate its performance on the testing set. As described in Table 3, we use two different configurations for training and testing. In the interpolation setting, for each maturity, we randomly select half of the contracts. As such,

Table 2: Daily statistics for January 2018.

	Mean	Std	Min	Max
Max moneyness	1.274	0.017	1.239	1.304
Min moneyness	0.186	0.019	0.174	0.241
Number of contracts	3434.000	193.934	3063.000	3747.000
Number of maturities	30.722	0.461	30.000	31.000

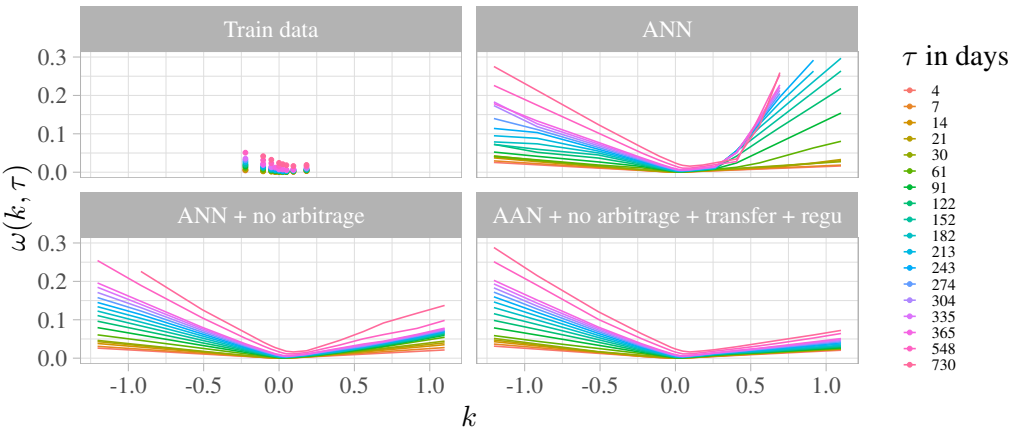

Figure 4: Extrapolation with different loss specification and weights initialization.

Table 3: Settings for the training and testing sets.

	Call Delta	Train/(Train + Test)
Interpolation	0–1	0.5
Extrapolation	0.02–0.98	≈ 0.5

we also sample options that are far out or in the money for training, and the testing error represents the approximation error for the range of moneyness that are actually observed. In the extrapolation setting, for each maturity, we select 70% of the contract that have an absolute Call option Delta in the range $[0.02, 0.98]$, see the Appendix B.2. This second filter is equivalent to selecting approximately 50% of all the contracts but with more observation around the money. Thus, we do not select options that are far out or in the money for training, and the testing error measures how well our model extrapolates. Finally, we again use the three set of conditions described in Table 1 to study how the arbitrage-related penalties affect the results.

In Table 4, we present our results. First, we describe the RMSE and MAPE. As expected, training errors are generally below testing errors. Somewhat surprisingly, extrapolation errors are most of the time smaller than interpolation errors. This may be because the surface is highly non-linear around the money, i.e. around $k = 0$, while the surface wings tend to be more linear. While MAPE errors might seem large, it should be noted that, for some contracts, the implied volatility is extremely small and thus even tiny deviations significantly impact this error measure. Interestingly, we see that the second condition (i.e., strong enforcement of the no-arbitrage constraints) also leads to the smallest interpolation and extrapolation errors. Furthermore, comparing the first and third conditions, we see that even a mild no-arbitrage constraint provides improvements over simply trying to minimize the RMSE and MAPE. Regarding CAL and BUT, we see that the models resulting from condition 1 and 2 are essentially arbitrage-free. As for the model resulting from no enforcement of the constraints (i.e., condition 3), it provides calendar-spread arbitrage opportunities both in interpolation and extrapolation, as well as butterfly arbitrage opportunities in extrapolation.

Time-series of the daily losses can be found in Appendix C.3.

6 Conclusion

We described a flexible methodology the price financial derivatives in an economically sensible way. This is achieved by modeling the implied volatility surface with a multilayer neural network and shaping it by penalizing the loss. We validate our approach with various numerical and empirical applications.

Table 4: Mean loss (with standard deviation). All numbers are multiplied by 100.

		Condition 1		Condition 2		Condition 3	
		Interpolation	Extrapolation	Interpolation	Extrapolation	Interpolation	Extrapolation
RMSE	train	3.12 (3.04)	4.79 (5.82)	2.47 (2.98)	1.56 (3.58)	5.46 (1.91)	8.1 (5.7)
	test	9.48 (7.52)	4.87 (5.91)	7.66 (6.95)	1.55 (3.47)	14.27 (4.67)	8.2 (5.75)
MAPE	train	16.04 (15.67)	16.39 (19.88)	12.81 (15.46)	5.42 (12.43)	27.96 (10.09)	27.66 (19.43)
	test	26.4 (22.65)	16.66 (20.11)	21.61 (21.91)	5.48 (12.26)	42.14 (13.54)	27.76 (19.45)
CAL	train	0.01 (0.02)	0.02 (0.03)	0.03 (0.07)	0.07 (0.18)	4.24 (2.75)	2.39 (1.78)
	test	0.01 (0.02)	0.02 (0.03)	0.03 (0.07)	0.07 (0.18)	4.24 (2.75)	2.39 (1.78)
BUT	train	0 (0)	0 (0)	0 (0)	0 (0)	0.86 (2.06)	3.36 (14.06)
	test	0 (0)	0 (0)	0 (0)	0 (0)	0.86 (2.06)	3.36 (14.06)

References

[1] *Abadi Martín, Barham Paul, Chen Jianmin, Chen Zhifeng, Davis Andy, Dean Jeffrey, Devin Matthieu, Ghemawat Sanjay, Irving Geoffrey, Isard Michael, others* . Tensorflow: A system for large-scale machine learning // 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). 2016. 265–283.

[2] *Ackerer Damien, Filipović Damir, Pulido Sergio*. The Jacobi stochastic volatility model // Finance and Stochastics. 2018. 22, 3. 667–700.

[3] *Barndorff-Nielsen Ole E*. Processes of normal inverse Gaussian type // Finance and stochastics. 1997. 2, 1. 41–68.

[4] *Bates David S*. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options // The Review of Financial Studies. 1996. 9, 1. 69–107.

[5] *Becker Sebastian, Cheridito Patrick, Jentzen Arnulf*. Deep optimal stopping // Journal of Machine Learning Research. 2019. 20, 74. 1–25.

[6] *Benaim Shalom, Friz Peter*. Regular variation and smile asymptotics // Mathematical Finance. 2009. 19, 1. 1–12.

[7] *Black Fischer, Scholes Myron*. The pricing of options and corporate liabilities // Journal of political economy. 1973. 81, 3. 637–654.

[8] *Buehler Hans, Gonon Lukas, Teichmann Josef, Wood Ben*. Deep hedging // Quantitative Finance. 2019. 1–21.

[9] *Cao Li-Juan, Tay Francis Eng Hock*. Support vector machine with adaptive parameters in financial time series forecasting // IEEE Transactions on neural networks. 2003. 14, 6. 1506–1518.

[10] *Carr Peter, Geman Hélyette, Madan Dilip B, Yor Marc*. The fine structure of asset returns: An empirical investigation // The Journal of Business. 2002. 75, 2. 305–332.

[11] *Carr Peter, Madan Dilip*. Option valuation using the fast Fourier transform // Journal of computational finance. 1999. 2, 4. 61–73.

[12] *Caruana Rich, Lawrence Steve, Giles C Lee*. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping // Advances in neural information processing systems. 2001. 402–408.

[13] *Corlay Sylvain*. B-spline techniques for volatility modeling // Working Paper. 2016.

[14] *Dugas Charles, Bengio Yoshua, Bélisle François, Nadeau Claude, Garcia René*. Incorporating second-order functional knowledge for better option pricing // Advances in neural information processing systems. 2001. 472–478.

[15] *El Euch Omar, Rosenbaum Mathieu*. The characteristic function of rough Heston models // Mathematical Finance. 2019. 29, 1. 3–38.

[16] *Fengler Matthias R*. Arbitrage-free smoothing of the implied volatility surface // Quantitative Finance. 2009. 9, 4. 417–428.

[17] *Fujii Masaaki, Takahashi Akihiko, Takahashi Masayuki*. Asymptotic Expansion as Prior Knowledge in Deep Learning Method for high dimensional BSDEs // Asia-Pacific Financial Markets. 2017. 1–18.

[18] *Gatheral Jim*. A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives // Presentation at Global Derivatives & Risk Management, Madrid. 2004.

[19] *Gatheral Jim, Jacquier Antoine*. Arbitrage-free SVI volatility surfaces // Quantitative Finance. 2014. 14, 1. 59–71.

[20] *Gençay Ramazan, Qi Min*. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging // IEEE Transactions on Neural Networks. 2001. 12, 4. 726–734.

[21] *Goodfellow Ian, Bengio Yoshua, Courville Aaron*. Deep learning. 2016.

[22] *Grasselli Martino*. The 4/2 stochastic volatility model: a unified approach for the Heston and the 3/2 model // Mathematical Finance. 2017. 27, 4. 1013–1034.

[23] *Hagan Martin T, Menhaj Mohammad B*. Training feedforward networks with the Marquardt algorithm // IEEE transactions on Neural Networks. 1994. 5, 6. 989–993.

[24] *Hagan Patrick S, Kumar Deep, Lesniewski Andrew S, Woodward Diana E*. Managing smile risk // The Best of Wilmott. 2002. 1. 249–296.

[25] *Han Jiequn, Jentzen Arnulf, Weinan E*. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning // arXiv preprint arXiv:1707.02568. 2017. 1–13.

[26] *Haykin Simon S, others* . Neural networks and learning machines/Simon Haykin. 2009.

[27] *Heaton JB, Polson NG, Witte JH*. Deep Learning in Finance // arXiv preprint arXiv:1602.06561. 2016.

[28] *Heaton JB, Polson NG, Witte Jan Hendrik*. Deep learning for finance: deep portfolios // Applied Stochastic Models in Business and Industry. 2017. 33, 1. 3–12.

[29] *Hernández-Lobato José Miguel, Hernández-Lobato Daniel, Suárez Alberto*. GARCH processes with non-parametric innovations for market risk estimation // International Conference on Artificial Neural Networks. 2007. 718–727.

[30] *Hernández-Lobato José Miguel, Lloyd James R, Hernández-Lobato Daniel*. Gaussian process conditional copulas with applications to financial time series // Advances in Neural Information Processing Systems. 2013. 1736–1744.

[31] *Heston Steven L.* A closed-form solution for options with stochastic volatility with applications to bond and currency options // The review of financial studies. 1993. 6, 2. 327–343.

[32] *Hinton Geoffrey, Vinyals Oriol, Dean Jeff.* Distilling the knowledge in a neural network // arXiv preprint arXiv:1503.02531. 2015.

[33] *Hornik Kurt*. Approximation capabilities of multilayer feedforward networks // Neural networks. 1991. 4, 2. 251–257.

[34] *Hornik Kurt, Stinchcombe Maxwell, White Halbert*. Multilayer feedforward networks are universal approximators // Neural networks. 1989. 2, 5. 359–366.

[35] *Hull John, White Alan*. The pricing of options on assets with stochastic volatilities // The journal of finance. 1987. 42, 2. 281–300.

[36] *Hull John C*. Options futures and other derivatives. 2017. 10.

[37] *Husken Michael, Goerick Christian*. Fast learning for problem classes using knowledge based network initialization // Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. 6. 2000. 619–624.

[38] *Hüsken Michael, Goerick Christian, Vogel Andrea*. Fast adaptation of the solution of differential equations to changing constraints // Proceedings of the second international ICSC Symposium on Neural Nomputation. 2000. 181–187.

[39] *Hutchinson James M, Lo Andrew W, Poggio Tomaso*. A nonparametric approach to pricing and hedging derivative securities via learning networks // The Journal of Finance. 1994. 49, 3. 851–889.

[40] *Itkin Andrey*. To sigmoid-based functional description of the volatility smile // The North American Journal of Economics and Finance. 2015. 31. 264–291.

[41] *Jaber Eduardo Abi, Larsson Martin, Pulido Sergio*. Affine volterra processes // arXiv preprint arXiv:1708.08796. 2017.

[42] *Kingma Diederik P, Ba Jimmy*. Adam: A method for stochastic optimization // ICLR. 2015.

[43] *Lee Roger W*. The moment formula for implied volatility at extreme strikes // Mathematical Finance. 2004. 14, 3. 469–480.

[44] *Leshno Moshe, Lin Vladimir Ya, Pinkus Allan, Schocken Shimon*. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function // Neural networks. 1993. 6, 6. 861–867.

[45] *Liu Shuaiqiang, Borovykh Anastasia, Grzelak Lech A, Oosterlee Cornelis W*. A neural network-based framework for financial model calibration // arXiv preprint arXiv:1904.10523. 2019.

[46] *Liu Shuaiqiang, Oosterlee Cornelis W., Bohte Sander M*. Pricing Options and Computing Implied Volatilities using Neural Networks // Risks. 2019. 7, 1.

[47] *Ludwig Markus*. Robust estimation of shape constrained state price density surfaces // The Journal of Derivatives. 2015. 22, 3. 56–72.

[48] *Madan Dilip B, Carr Peter P, Chang Eric C*. The variance gamma process and option pricing // Review of Finance. 1998. 2, 1. 79–105.

[49] *Márquez-Neila Pablo, Salzmann Mathieu, Fua Pascal*. Imposing hard constraints on deep networks: Promises and limitations // arXiv preprint arXiv:1706.02025. 2017.

[50] *Merton Robert C*. Theory of Rational Option Pricing // The Bell Journal of Economics and Management Science. 1973. 4, 1. 141–183.

[51] *Merton Robert C*. Option pricing when underlying stock returns are discontinuous // Journal of financial economics. 1976. 3, 1-2. 125–144.

[52] *Prechelt Lutz*. Early stopping-but when? // Neural Networks: Tricks of the trade. 1998. 55–69.

[53] *Roper Michael*. Arbitrage free implied volatility surfaces // preprint. 2010.

[54] *Sirignano Justin, Spiliopoulos Konstantinos*. DGM: A deep learning algorithm for solving partial differential equations // Journal of Computational Physics. 2018. 375. 1339–1364.

[55] *Stein Elias M, Stein Jeremy C*. Stock price distributions with stochastic volatility: An analytic approach // The Review of Financial Studies. 1991. 4, 4. 727–752.

[56] *Sutskever Ilya, Martens James, Dahl George, Hinton Geoffrey*. On the importance of initialization and momentum in deep learning // International conference on machine learning. 2013. 1139–1147.

[57] *Yang Yongxin, Zheng Yu, Hospedales Timothy M*. Gated neural networks for option pricing: Rationality by design // Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[58] *Zheng Yu*. Machine learning and option implied information. 2018.

Online Appendix

A Literature review

The assumption of constant volatility in the Black-Scholes-Merton model has long been challenged empirically and various stochastic volatility models have been developed to tackle its limitations. Some examples are the Hull-White [35], the Stein-Stein [55], the Heston [31], the Variance-Gamma [48], the normal inverse Gaussian [3], the CGMY [10], the 4/2 [22], the Jacobi [2], rough Heston [15], and affine Volterra [41] models.

Albeit the development of more flexible models for stock prices, their statistical flexibility remained limited and they may be computationally too costly to calibrate for some real-world applications. For these reasons, parametric and nonparametric approaches have been developed aiming to interpolate, and sometimes to extrapolate, the implied volatility surface. These approaches includes the stochastic volatility inspired of [18, 19], and the smoothing spline techniques of [16, 13], among many others.

Several shallow neural networks approaches have also been developed to smooth option prices directly. [14] constructed a one hidden layer neural network monotonic or convex in its input coordinate, and taking only positive values. However, the construction is specific rendering it impossible to extend to multilayer neural networks, it performs poorly with both short and long maturities, and do not prevent all forms of static arbitrage opportunities. Recently, this model has been extended in a PhD thesis [58] by adding a gated unit layer linking the input to multiple models à la Dugas. [47] proposes a one hidden layer approach to model the implied total variance and his approach includes multiple ad-hoc rules. For examples, the extrapolation behavior to unobserved areas of the implied volatility surface is controlled for by adding discretionary data points, the training procedure is restarted until 25 neural nets are found to be arbitrage-free at selected strikes and maturities, the final implied volatility surface is obtained by aggregating over the best three models, and so on. The sigmoid-based approach of [40] to model the implied volatility smile is closely related to a neural network approach.

On a broader note, the financial applications of neural networks are booming as a consequence of the progress made in deep learning and of the availability of specialized software and hardware. They have for examples been used in [45, 46] to speed-up the pricing and calibration of options in stochastic volatility models, and in [8] to approximate optimal but intractable option hedging strategies with market frictions.

B Baseline models

B.1 Stochastic volatility models

The Bates model [4] is a combination of the Merton jump diffusion model [51] and the Heston stochastic volatility model [31]. The stock price dynamics is given by

$$\begin{aligned} dS_t/S_t &= (r-\delta)dt + \sqrt{V_t}dW_t^1 + dN_t \\ dV_t &= \kappa(\theta - V_t)dt + \sigma\sqrt{V_t}dW_t^2 \end{aligned}$$

where r is the interest rate, δ is the dividend yield, V_t is the spot volatility, θ is the long-run volatility, κ is the speed of mean-reversion, σ is the volatility of volatility, and W_t^1 and W_t^2 are two correlated Brownian motion with parameter ρ. The process N_t is a compound Poisson process with intensity λ and independent jumps J with

$$\ln(1+J) \sim \mathcal{N}\left(\ln(1+\beta) - \frac{1}{2}\alpha^2,\, \alpha^2\right)$$

where the parameters α and β determine the distribution of the jumps, and the Poisson process is assumed to be independent of the Brownian motions. The Heston model is retrieved by removing the jump component dN_t from the Bates model.

As the characteristic function of the log-price is known, we used the Fast Fourier transform method [11] in order to compute option prices efficiently.

B.2 Option Greeks

The option Delta Δ and Vega ν are the sensibilities of the option price, in the Black-Scholes formula, with respect to changes in the underlying price and in the volatility parameter respectively. They are given by

$$\Delta = \begin{cases} \mathrm{e}^{-\delta\tau}\Phi(d_+) & \text{for Calls} \\ -e^{-\delta\tau}\Phi(-d_+) & \text{for Puts} \end{cases}$$

and by

$$\nu = S\mathrm{e}^{-\delta\tau}\phi(d_+)\sqrt{\tau} = Ke^{-r\tau}\phi(d_-)\sqrt{\tau}$$

for both Calls and Puts, where Φ and ϕ denotes respectively the standard Gaussian CDF and PDF.

B.3 Calibration to market prices

We calibrate the baseline models on out-of-the-money options by minimizing the Vega weighted price errors, which approximates the implied volatility errors. By doing so, we avoid inverting the Black-Scholes formula to retrieve the implied volatility values and thus we speed-up the calibration procedure. The following procedure is valid only for European-style options, meaning options that can only be exercised at maturity, which is the typical style for options on financial indices such as the S&P 500.

With a linear regression we extract from the put-call parity the risk-free rate r and the dividend yield δ for each maturity. We denote here π_j, σ_j, and ν_j the j-th option price, implied volatility, and Vega. Similarly $\hat{\pi}_j$ and $\hat{\sigma}_j$ denote the model option price and implied volatility. We calibrate the models by minimizing the Vega-weighted root-mean-square-error (RMSE)

$$\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{\pi_j - \hat{\pi}_j}{\nu_j}\right)^2}$$

where N is the number of out-of-the-money options on a particular day. This criterion is a computationally efficient approximation for the implied volatility surface RMSE criterion which follows by observing that

$$\sigma_j - \hat{\sigma}_j \approx \frac{\pi_j - \hat{\pi}_j}{\nu_j} \quad \text{when} \quad \pi_j \approx \hat{\pi}_j.$$

C Additional information and results

C.1 Model flexibility

The first row in Figure 5 displays implied variance slices for the Black-Scholes, Bates, and a sample extracted from S&P 500 option mid-prices on October 10, 2018. The second row in Figure 5 displays the corresponding ANN model fit with 4 layers, 20 neurons per layer, and without arbitrage constraints $\lambda_1 = \lambda_2 = 0$. The learning algorithm described in B.3 was used. This ANN model is sufficiently flexible to reproduce the different implied volatility surface shapes.

C.2 Results for the Heston model

Figures 6 and 7 display the number of iteration before the calibration stops and, respectively, the different loss terms for ANN models fitted the Heston sample.

C.3 Results for time series

Figure 8 displays the loss terms time series for the ANN models fitted on real data.

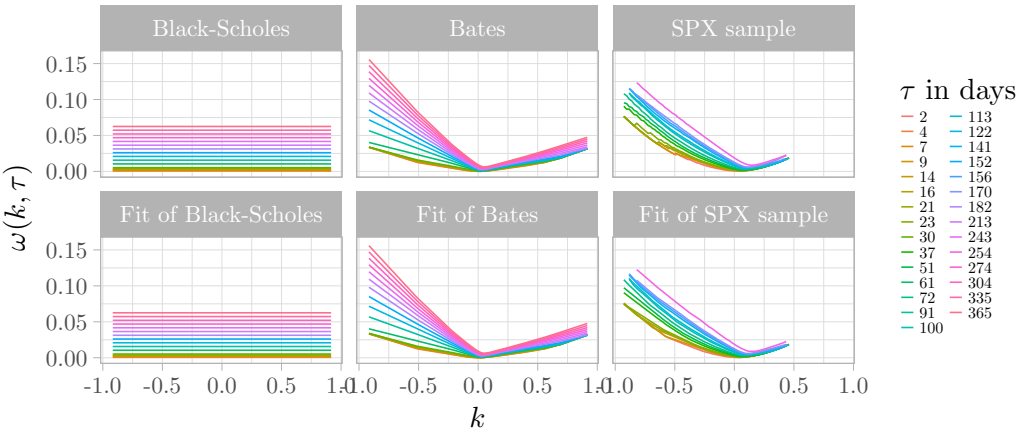

Figure 5: Implied variance slices and the corresponding fit with an ANN model with Condition 3.

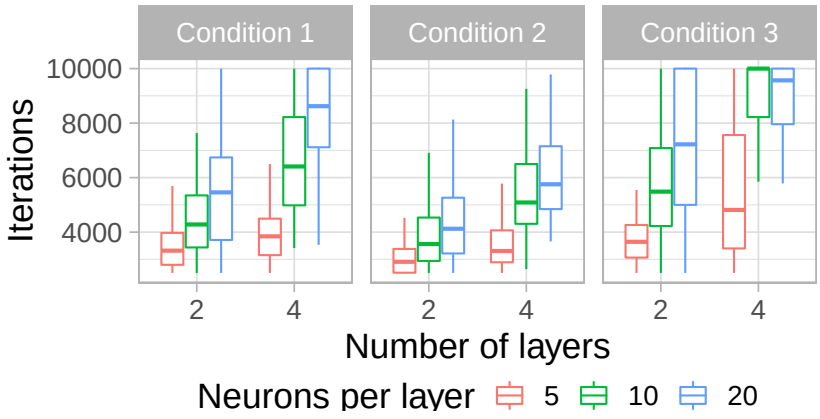

Figure 6: Number of iterations before the calibration stops for different combination of number of layers and neurons per layer. Synthetic data is generated from the Heston model.

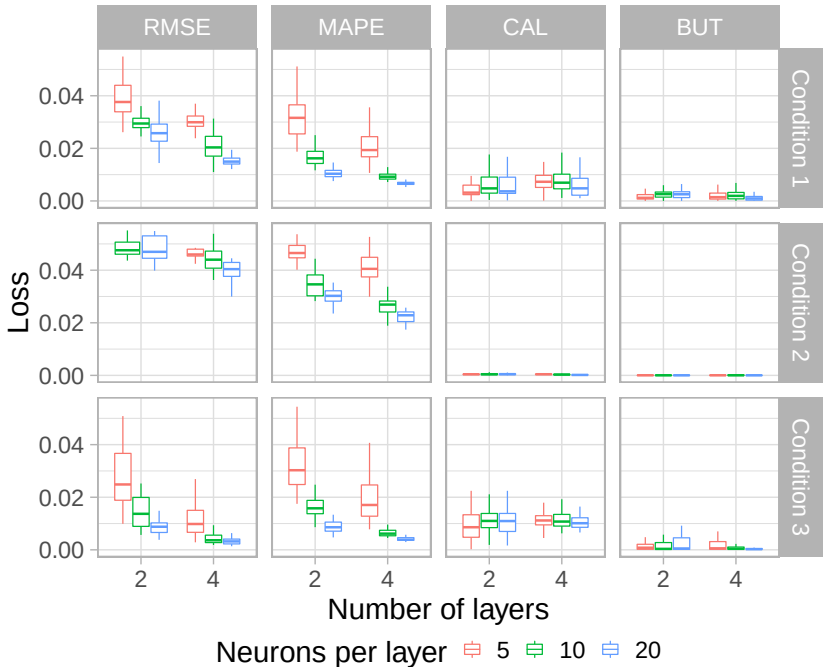

Figure 7: Losses for different combination of number of layers and neurons per layer. Synthetic data is generated from the Heston model.

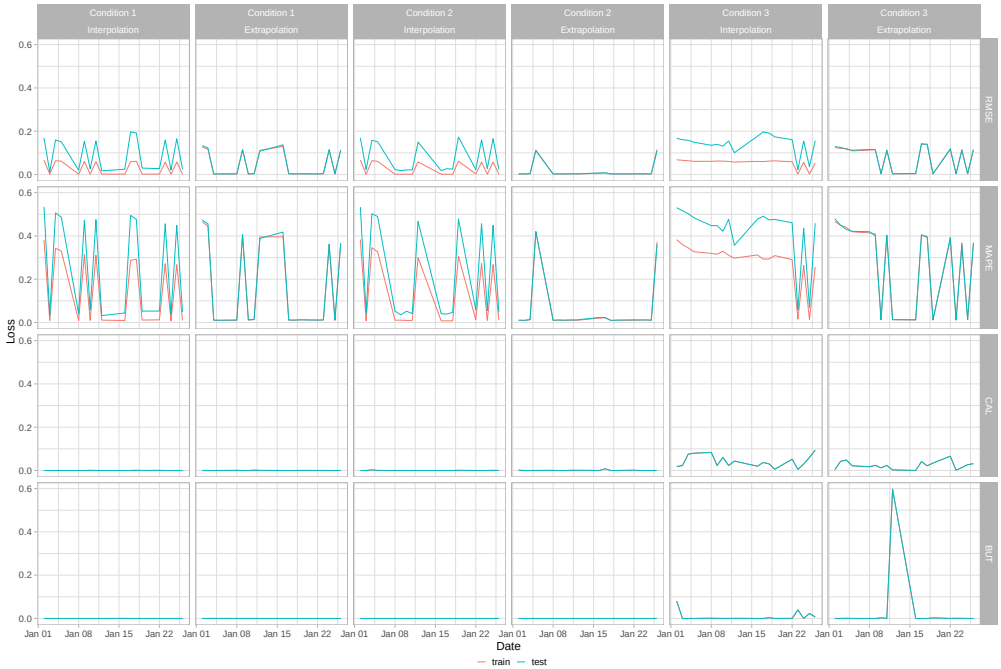

Figure 8: Losses for the fitted model in January 2018.