Deep Local Volatility

Marc Chataigner* Stéphane Crépey* Matthew Dixon†

April 29, 2020

Abstract

Deep learning for option pricing has emerged as a novel methodology for fast computations with applications in calibration and computation of Greeks. However, most of these approaches do not enforce any no-arbitrage conditions. In this article, we develop a deep learning approach for no-arbitrage interpolation of European vanilla option prices. In particular, we demonstrate the modification of the standard deep learning methodology to enforce the no-arbitrage constraint and we specify the experimental design parameters that are needed for adequate performance. A novel component is the use of the Dupire formula to enforce bounds on the local volatility associated with (non-arbitrable) option prices, during the network fitting. Numerical results[1] on real datasets of DAX vanilla options demonstrate the numerical error in the price, implied volatility and local volatility surface.

Option pricing; Deep learning; No-Arbitrage; Local Volatility.
91G20, 62M45, 91G60.

1 Introduction

A recent approach to option pricing involves reformulating the pricing problem as a surrogate modeling problem. Once reformulated, the problem is amenable to standard supervised machine learning methods such as Neural networks and Gaussian processes. This is particularly suitable in situations with a need for fast computations and a tolerance to in-exact approximation.

In their seminal paper, Hutchinson et al. (1994) use a radial basis function neural network for delta-hedging. Their network is trained to Black-Scholes prices, using the time-to-maturity and the moneyness as input variables, without shape constraints. They use the hedging performance of the ensuing delta-hedging strategy as a performance criterion.

*Department of Mathematics, University of Evry, Paris Saclay; stephane.crepey@univ-evry.fr, marc.chataigner@univ-evry.fr. The PhD thesis of Marc Chataigner is co-funded by the Research Initiative "Modélisation des marchés actions, obligations et dérivés", financed by HSBC France under the aegis of the Europlace Institute of Finance, and by a public grant as part of investissement d'avenir project, reference ANR-11-LABX-0056-LLH LabEx LMH. The views and opinions expressed in this paper are those of the authors alone and do not necessarily reflect the views or policies of HSBC Investment Bank, its subsidiaries or affiliates.

†Department of Applied Mathematics, Illinois Institute of Technology, Chicago; matthew.dixon@iit.edu.

[1] A Python notebook, compatible with Google Colab, and accompanying data are available in https://github.com/mChataign/DupireNN. Due to file size constraints, the notebook must be run to reproduce the figures and results in this article.

Garcia and Gençay (2000) and Gençay and Qi (2001) improve the stability of the previous approach by adding the outputs of two such neural networks, weighted by respective moneyness and time-to-maturity functionals. Dugas et al. (2009) introduce the first neural network architecture guaranteeing arbitrage-free vanilla option pricing. However, Ackerer et al. (2019) show that the corresponding hard constrained networks are very difficult to train in practice. Instead, they advocate the learning of the implied volatility (rather than the prices) by a standard feedforward neural network with soft-constraints, i.e. regularization, which penalizes calendar spread and butterfly arbitrages. The call and put prices must also be decreasing and increasing by strike respectively.

We propose simple neural network architectures and training methodology which satisfy these shape constraints. In contrast to Dugas et al. (2009), following Ackerer et al. (2019), we also explore soft constraints alternatives to hard constraints in the network configuration, due to the loss of representation power of the latter. However, our networks are trained to prices, versus implied volatilities in Ackerer et al. (2019). Moreover, a novel aspect of our approach is to focus on the associated local volatility surface, considered both for itself and as a penalization device in our soft constraints approach.

2 Problem Statement

We consider European vanilla option prices on a stock or index S. We assume that a deterministic short interest rate term structure $r(t)$ of the corresponding economy has been bootstrapped from the its zero coupon curve, and that a term structure of deterministic continuous-dividend-yields $q(t)$ on S has then been extracted from the prices of the forward contracts on S. Without restriction given the call-put parity relationship, we only consider put option prices. We denote by $P^\star(T,K)$ the market price of the put option with maturity T and strike K on S, observed for a finite number of pairs (T,K) at a given day.

Our goal is to construct a nonarbitrable put price surface $P : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ in $\mathcal{C}^{1,2}(\mathbb{R}_+^\star \times \mathbb{R}_+^\star) \cap \mathcal{C}^0(\mathbb{R}_+ \times \mathbb{R}_+^\star)$, interpolating $P^\star$ up to some error term. As is well known (see e.g. Section 9.2.1 in Crépey (2013)), the corresponding local volatility surface, say $\sigma(\cdot,\cdot)$, is given by the Dupire (1994) formula

$$\frac{\sigma^2(T,K)}{2} = \frac{\partial_T P(T,K) + (r-q)\partial_K P(T,K) + qP(T,K)}{K^2 \partial^2_{K^2} P(T,K)}.$$

In terms of $p(T,k) = \exp\left(\int_0^T q(t)dt\right)P(T,K)$, where $k = K\exp\left(\int_0^T (r(t)-q(t))dt\right)$, the formula reads

$$\frac{\sigma^2(T,K)}{2} = \frac{\partial_T p(T,k)}{k^2 \partial^2_{k^2} p(T,k)}. \tag{1}$$

Given a rectangular domain of interest in maturity and strike, We rescale further the inputs, $T' = T/(max(T) - min(T))$ and $k' = k/(max(k) - min(k)$, so that the domain of the pricing map is $\Omega := [0,1]^2$. For ease of notation, we shall hereon drop the superscript. This rescaling avoids any one independent variable dominating over another during the fitting. We then learn the modified market prices $p^\star = p^\star(T,k)$.

For the Dupire formula (1) to be meaningful, its output must be nonnegative. This holds, in particular, whenever the interpolating map p exhibits nonnegative derivatives w.r.t. T and second derivative w.r.t. k, which corresponds to (static) nonarbitrage relationships on the option prices surface. In both networks considered in the paper, these derivatives are

available analytically via the neural network automatic differentiation features. Hard or soft (regularization) shape constraints can be used to enforce these shape properties, exactly in the case of hard constraints and approximately in the case of soft constraints.

3 Shape Preserving Neural Networks

We consider parameterized maps $p = p_{\mathbf{W},\mathbf{b}}$

$$(T,k) \ni \mathbb{R}^2_+ \xrightarrow{p} p_{\mathbf{W},\mathbf{b}}(T,k) \in \mathbb{R}_+,$$

given as deep neural networks with two hidden layers. As detailed in Goodfellow et al. (2016), these take the form of a composition of simpler functions:

$$p_{\mathbf{W},\mathbf{b}}(x) = f^{(3)}_{W^{(3)},b^{(3)}} \circ f^{(2)}_{W^{(2)},b^{(2)}} \circ f^{(1)}_{W^{(1)},b^{1)}}(x),$$

where

$$\mathbf{W} = (W^{(1)}, W^{(2)}, W^{(3)}) \text{ and } \mathbf{b} = (b^{(1)}, b^{(2)}, b^{(3)})$$

are weight matrices and bias vectors, and the $f^{(l)} := \varsigma^{(l)}(W^{(l)}x + b^{(l)})$ are semi-affine, for nondecreasing activation functions $\varsigma^{(l)}$ applied to their (vector-valued) argument componentwise. Any weight matrix $W^{(\ell)} \in \mathbb{R}^{m\times n}$ can be expressed as an n column $W^{(\ell)} = [\mathbf{w}^{(\ell)}_1, \ldots, \mathbf{w}^{(\ell)}_n]$ of m-vectors, for successively chained pairs (n,m) of dimensions varying with $l = 1,2,3$ (starting from $n = 2$, the number of inputs, for $l = 1$, and ending up with $m = 1$, the number of outputs, for $l = 3$).

In the hard constraints case, our network is sparsely connected in the sense that, with $x = (T,k)$ as above,

$$f^{(1)}_{W^{(1)},b^{(1)}}(x) = [f^{(1,T)}_{W^{(1,T)},b^{(1,T)}}(T), f^{(1,k)}_{W^{(1,k)},b^{(1,k)}}(k)],$$

where $W^{(1,T)}, b^{(1,T)}$ and $W^{(1,k)}, b^{(1,k)}$ correspond to parameters of sub-graphs for each input, and

$$f^{(1,T)}(T) := \varsigma^{(1,T)}(W^{(1,T)}T + b^{(1,T)}),\ f^{(1,k)}(k) := \varsigma^{(1,k)}(W^{(1,k)}k + b^{(1,k)}).$$

To impose the shape constraints relevant for put options, it is then enough to restrict ourselves to nonnegative weights, and to convex (and nondecreasing) activation functions, namely

$$\text{softplus}(x) := \ln(1 + e^x),$$

except for $\varsigma^{(1,T)}$, which will be taken as an S-shaped sigmoid $(1 + e^{-x})^{-1}$.

Hence, the network is nondecreasing (but not necessarily convex) in T, and convex and nondecreasing in k, as a composition (restricted to the k variable) of convex and nondecreasing functions of k. Figure 1 illustrates the shape preserving feed forward architecture with two hidden layers containing 10 hidden nodes. For avoidance of doubt, the figure is not representative of the number of hidden neurons used in our experiments. However, the connectivity is representative. The first input variable, T, is only connected to the first 5 hidden nodes and the second input variable, k, is only connect to the last 5 hidden nodes. Effectively, two sub-networks have been created where no information from the input layer crosses the sub-networks until the second hidden layer. In other words, each sub-network is a function of only one input variable. This property is the key to imposing different hard shape constraints w.r.t. each input variable.

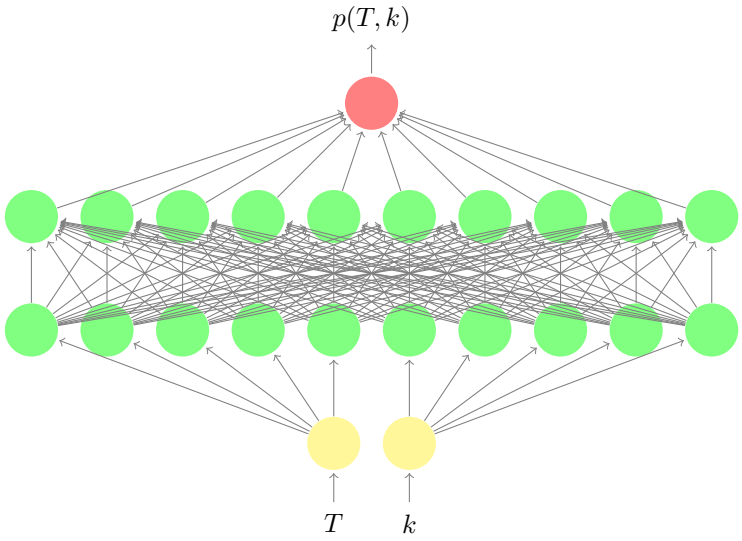


Figure 1: *A shape preserving (sparse) feed forward architecture with two hidden layers containing 10 hidden nodes. The first input variable, T, is only connected to the 5 left most hidden nodes in the first hidden layer and the second input variable, k, is only connected to the 5 right most hidden nodes in the first hidden.*

3.1 Approximation error bound

We recall a result from Ohn and Kim (2019) which describes how the sparsity in a neural network affects its approximation error.

Let us denote the network parameters $\theta := (\mathbf{W}, \mathbf{b})$. We define the network parameter space in terms of the layers depth L and width N, and numbers of inputs and outputs,

$$\Theta_{i,o}(L, N) := \{\theta : L(\theta) \leq L, n_{max}(\theta) \leq N, in(\theta) = i, out(\theta) = o\}.$$

We also define the restricted parameter space

$$\Theta_{i,o}(L, N, \Sigma, B) := \{\theta \in \Theta_{i,o}(L, N) : |\theta|_0 \leq \Sigma, |\theta|_\infty \leq B\},$$

where $|\theta|_0$ is the number of nonzero components in θ and $|\theta|_\infty$ is the largest absolute value of elements of θ. Let the activation ς be either piecewise continuous or locally quadratic[2]. For example, softplus and sigmoid functions are locally quadratic. Let the function being approximated, $p \in \mathcal{H}^{\alpha,R}([0,1]^i)$ be Hölder smooth with parameters $\alpha > 0$ and $R > 0$, where $\mathcal{H}^{\alpha,R}(\Omega) := \{p \in \mathcal{H}^\alpha(\Omega) : ||p||_{\mathcal{H}^\alpha(\Omega)} \leq R\}$. Then Theorem 4.1 in Ohn and Kim (2019) states the existence of positive constants L_0, N_0, Σ_0, B_0 depending only on i, α, R and ς s.t. for any $\epsilon > 0$, the neural network

$$\theta_\epsilon \in \Theta_{i,1}\left(L_0 log(1/\epsilon), N_0 \epsilon^{-i/\alpha}, \Sigma_0 \epsilon^{-i/\alpha} log(1/\epsilon), B_0 \epsilon^{-4(i/\alpha+1)}\right)$$

satisfies $\sup_{x\in[0,1]^i} |p(x) - p_{\theta_\epsilon}(x)| \leq \epsilon$. Figure 2 shows the upper bound Σ on the network sparsity, $|\theta|_0 \leq \Sigma$, as a function of the error tolerance ϵ and Hölder smoothness, α, of the

[2]A function $\varsigma : \mathbb{R} \to \mathbb{R}$ is locally quadratic if $\exists$ an open interval $(a, b) \subset \mathbb{R}$ over which ς is three times continuously differentiable with bounded derivatives and $\exists t \in (a, b)$ s.t. $\varsigma'(t) \neq 0$ and $\varsigma''(t) \neq 0$.

function being approximated. Keeping the number of neurons in each layer fixed, the graph shows that a denser network, with a higher upper bound, results in a lower approximation error. Conversely, networks with a low number of non-zero parameters, due to zero weight edges, exhibit larger approximation error. In the context of no-arbitrage pricing, the theorem suggests a tradeoff between using a sparse network to enforce the shape constraints, yet increasing the approximation error. The adverse effect of using a sparse network should also diminish with increasing smoothness in the function being approximated.

3.2 Training

To fit our fully connected or sparse networks to the grid of option market prices, we solve a loss minimization problem using observations $\{x_i = (T_i, k_i), p^\star(x_i)\}_{i=1}^n$ of n maturity-strike pairs and the corresponding market put prices:

$$\min_{\mathbf{W},\mathbf{b}} \ \frac{1}{n}\sum_{i=1}^{n}\Big(|p^\star(x_i) - p(x_i)| + \lambda^\mathsf{T}\phi(x_i)\Big), \tag{2}$$

where $p = p_{\mathbf{W},\mathbf{b}}$ and $\phi = \phi_{\mathbf{W},\mathbf{b}}$ is a regularization penalty vector:

$$\phi := [(\partial_T p)^-, (\partial^2_{k^2} p)^-, (dup(p) - \overline{a})^+ + (dup(p) - \underline{a})^-],$$

and $dup(p)$ corresponds to the right-hand-side in (1). The choice to measure the error $p^\star - p$ under the L_1 norm, rather than L_2 norm, in (2) is motivated by a need to avoid allocating too much weight to the deepest in-the-money options. The loss function is non-convex, possessing many local minima and it is generally difficult to find a global minimum. The first two elements in the penalty vector enforce the shape constraints and the third element enforces bounds on the estimated local half-variance, $dup(p)$, where constants (which could also be functions of time) $0 < \underline{a} < \overline{a}$ denote related desired lower and upper bounds. Of course, as soon as penalizations are used, a further difficulty, typically involving grid search, is the need to determine suitable values of the corresponding "Lagrange multipliers" $\lambda \in \mathbb{R}^3_+$, ensuring the right balance between fit to the market prices and the targeted constraints.

4 Experimental Design

In this section we benchmark four different combinations of architecture and optimization criteria. In each case the error between the prices of the calibrated model and the market data are evaluated on both the training and an out-of-sample test set. Unless reported otherwise, all numerical results shown below correspond to test sets.

Note that only the "hard constraints" approach theoretically guarantees that the associated Dupire formula (1) returns a positive function. While soft constraints reduce the risk of statistical arbitrage (in the sense of mismatch between model and market prices), they do not however fully prevent butterfly or calendar spread type arbitrages. In particular, the penalties only control the corresponding derivatives at the training points. Compliance with the arbitration constraints on the majority of the points in the test set is due only to the regularity of these derivatives.

The training and test sets are prepared using daily datasets of DAX index European vanilla options of different available strikes and maturities, listed on the 7^{th}, 8^{th} (by default below), and 9^{th}, August 2001, i.e. same datasets as already used in previous work Crépey

(2002, 2004), for comparison purposes. The corresponding values of the underlying are $S = 5752.51, 5614.51$ and 5512.28. Each set of observations has just under 200 options market prices. The associated interest rate and dividend yield curves are constructed from zero-coupon and forward curves, themselves obtained from quotations of standard fixed income linear instruments and from call/put parity applied to the option market prices.

Each network has two hidden layers, each with 200 neurons per hidden layer. Note that Dugas et al. (2009) only uses one hidden layer. Two was found important in practice in our case. All networks are fitted with an ADAM optimizer. The learning rate is divided by 10 whenever no improvement in the error on the training set is observed during 100 consecutive epochs. The total number of epoches is limited to $10,000$.

After training, a local volatility surface is extracted from the interpolated prices by application of the Dupire formula (1), leveraging on the availability of the corresponding exact sensitivities. This local volatility surface is then compared with the one calibrated by a classical nonlinear least squares optimization procedure with Tikhonov regularization (see Crépey (2002)).

5 Numerical Results

Table 1 shows the pricing RMSEs for four different combinations of architecture and optimization criteria. For the sparse network with hard constraints, we set $\lambda = \mathbf{0}$. For the sparse and dense networks with soft constraints (i.e. penalization but without the conditions on the weights of Section 3), we set $\lambda = [1.0 \times 10^5, 1.0 \times 10^3, 0]$. For avoidance of doubt, so far we do not impose local volatility bounds.

The sparse network with hard constraints is observed to exhibit significant pricing error, which suggests that this approach is too limited in practice to approach market prices. This conclusion is consistent with Ackerer et al. (2019), who choose a soft-constraints approach in the implied volatility approximation (in contrast to our approach which approximates prices).

	Sparse network		Dense network	
	Hard constraints	Soft constraints	Soft constraints	No constraints
Training dataset	28.13	6.87	2.28	2.56
Testing dataset	28.91	4.09	3.53	3.77
Indicative training times	200s	400s	200s	120s

Table 1: *Pricing RMSE (absolute pricing errors) without local volatility constraints.*

Figure 3 compares the percentage errors in implied volatilities using the sparse network with hard constraints and the dense network with soft constraints approaches, corresponding to the columns 1 and 3 of Table 1. Errors are capped at 10% so as not to overwrite the scale of the figures. This confirms that the error levels of the hard constraints approach are too high to imagine a practical use of this approach: the corresponding model would be immediately arbitrable in relation to the market. Those of the soft constraint approach are much more acceptable, with high errors confined to short maturities or far from the money, i.e. in the region where prices provide little information on volatility.

5.1 Local volatility

We now introduce local volatility bounds into the regularization to improve the overall fit in prices and stabilize the local volatility surface. Table 2 shows the RMSEs in absolute pricing resulting from repeating the same set of experiments reported in Table 1, but with the local volatility bounds included in the regularization. For the sparse network with hard constraints, we set $\lambda = [0, 0, 10]$ and choose $\underline{a} = 0.05^2/2$ and $\overline{a} = 0.4^2/2$. For the sparse and dense networks with soft constraints, we set $\lambda = [1.0 \times 10^5, 1.0 \times 10^3, 10]$. Compared to Table 1, we observe improvement in the test error for the hard and soft constraints approaches when including the additional local volatility penalty term.

	Sparse network		Dense network	
	Hard constraints	Soft constraints	Soft constraints	No constraints
Training dataset	28.04	3.44	2.48	3.48
Testing dataset	27.07	3.33	3.36	4.31
Indicative training times	400s	600s	300s	250s

Table 2: *Price RMSE (absolute pricing errors) with local volatility bounds in the regularization.*

Figure 4 shows the comparison of the local volatility surfaces obtained using hard constraints without local volatility bounds, soft constraints without and with local volatility bounds, as well as the Tikhonov regularization approach of Crépey (2002), on data listed on August 7^{th}, 8^{th}, and 9^{th}, 2001. The hard constraint approach yields the smoothest local volatility surface, although the shape is not comparable to those using the Tikhonov regularization approach of Crépey (2002). The soft constraint approach without the local volatility constraints is both spiky (exhibiting outliers on a given day) and unstable (from day to day). In contrast, the soft constraint approach with local volatility bounds yields a reasonable local volatility surface, both at fixed calendar time and in terms of stability across calendar time. From this point of view the results are then rather comparable to those obtained by Tikhonov regularization (which takes of the order of 30s to run).

While the neural network training times are longer than the numerical optimization involved in the Tikhonov regularization method, the neural network provides the full surface of prices and local volatilities, as opposed to just values at the nodes of a trinomial tree under the approach of Crépey (2002).

6 Conclusion

Deep learning for option pricing has emerged as a novel computational approach for fast option pricing and Greeking. We introduced three variations of deep learning methodology to enforce no-arbitrage interpolation of European option prices: (i) modification of the network architecture to embed shape constraints (hard constraints), (ii) use of shape regularization to enforce the constraints (soft constraints), and (iii) additional use of local volatility bounds in the regularization via the Dupire formula.

Our experimental results suggest that hard constraints reduce the representational power of the network, yet provide the only fail-safe approach to no-arbitrage approximation. Future research would be required to actually prove loss of universal representation in feed-forward networks in the presence of constraints on weights.

On the other-hand, soft constraints provide much more accurate prices and implied volatilities. If the Dupire formula is included in the regularization, the corresponding local volatility surface is also reasonably regular (at fixed day) and stable (from day to day). The neural network training times are longer than the numerical optimization involved in the Tikhonov regularization method. But neural networks estimate the full surface of no-arbitrage prices and local volatilities, as opposed to just values at the nodes of a trinomial tree under the approach of Crépey (2002). This is a material advantage for the use of the local volatility in production.

References

Ackerer, D., N. Tagasovska, and T. Vatter (2019). Deep smoothing of the implied volatility surface. *Available at SSRN 3402942*.

Crépey, S. (2002). Calibration of the local volatility in a trinomial tree using Tikhonov regularization. *Inverse Problems 19*(1), 91.

Crépey, S. (2004). Delta-hedging vega risk? *Quantitative Finance 4*(5), 559–579.

Crépey, S. (2013). *Financial Modeling: A Backward Stochastic Differential Equations Perspective*. Springer Finance Textbooks.

Dugas, C., Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia (2009). Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research 10*(Jun), 1239–1262.

Dupire, B. (1994). Pricing with a smile. *Risk 7*, 18–20.

Garcia, R. and R. Gençay (2000). Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics 94*(1-2), 93–115.

Gençay, R. and M. Qi (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks 12*(4), 726–734.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press.

Hutchinson, J. M., A. W. Lo, and T. Poggio (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance 49*(3), 851–889.

Ohn, I. and Y. Kim (2019, June). Smooth Function Approximation by Deep Neural Networks with General Activation Functions. *Entropy 21*(7), 627.

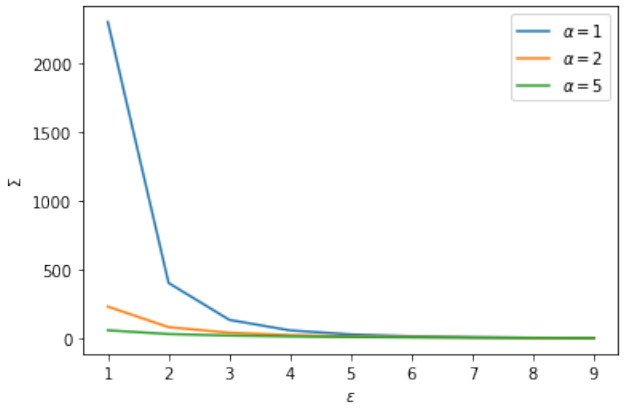


Figure 2: *The upper bound on the network sparsity,* $|\theta|_0 \leq \Sigma$*, as a function of the error tolerance* ϵ *and Hölder smoothness,* α*, of the function being approximated. As* α *increases, the value of* Σ *is observed to decrease. The plot is shown for* $i = 2$ *and* $\Sigma_0 = 10$*.*

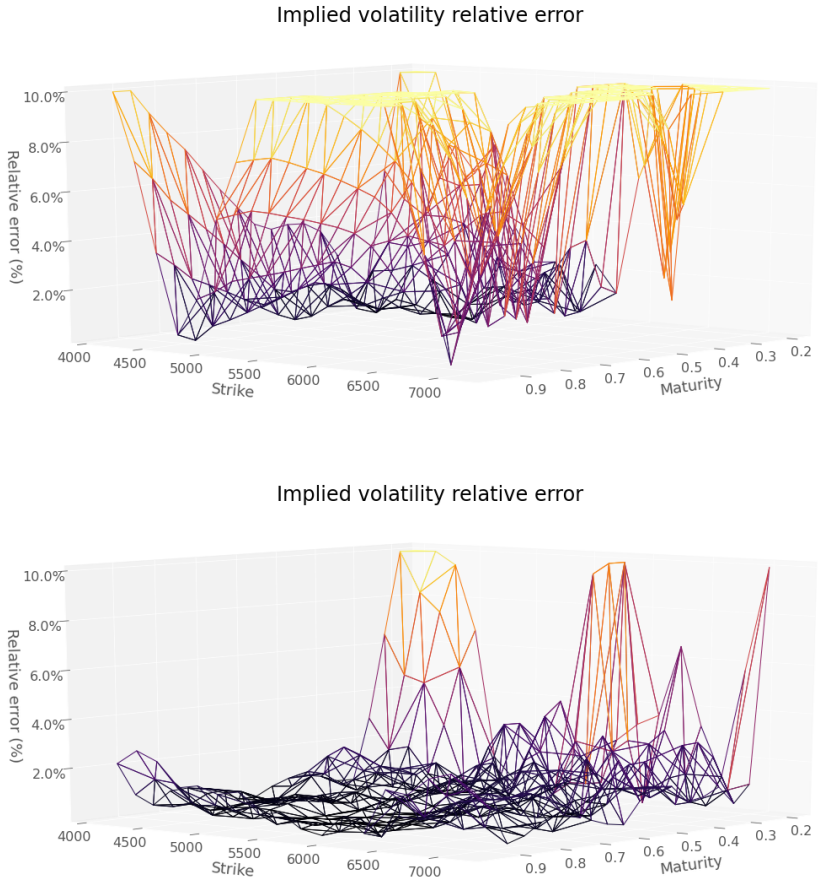
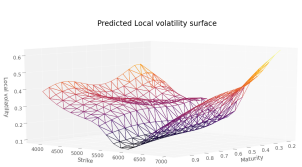
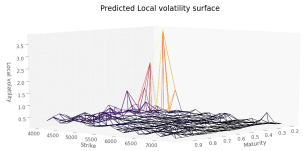


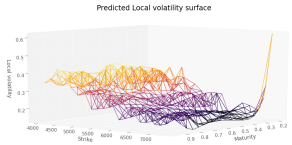
Figure 3: *Percentage relative error in the implied volatilities using (a) hard constraints (b) dense networks with soft constraints.*



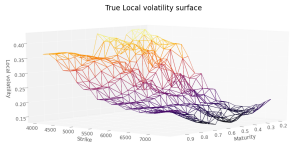
(a) Hard Constraints



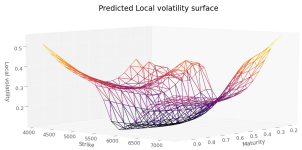
(b) Soft constraints (w/o local vol. constraints)



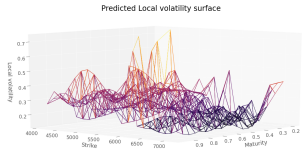
(c) Soft constraints (with local vol. constraints)



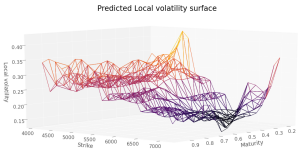
(d) Tikhonov local volatility



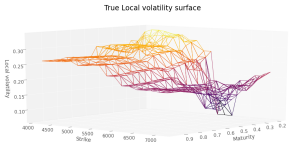
(e) Hard Constraints



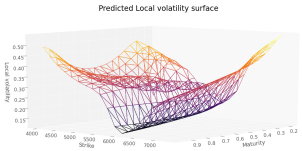
(f) Soft constraints (w/o local vol. constraints)



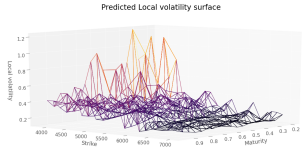
(g) Soft constraints (with local vol. constraints)



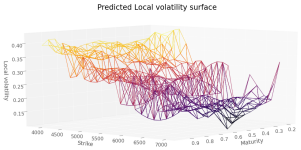
(h) Tikhonov local volatility



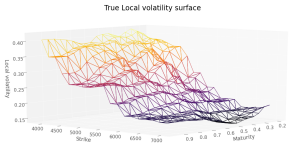
(i) Hard Constraints



(j) Soft constraints (w/o local vol. constraints)



(k) Soft constraints (with local vol. constraints)



(l) Tikhonov local volatility

Figure 4: *Local volatility over three consecutive days (each row corresponds to a day).*