Machine Learning in Finance

Robust Pricing and Hedging via Neural SDEs: a tutorial

MASSIN Keryann, LIFFRAN Anna

May 1, 2024

FERMANIAN Jean-David, PHAM Huyên

Contents

1 Introduction

In market finance, numerous challenges arise when attempting to efficiently determine the fair price of an asset or derivative. Within the pricing and hedging framework, various strategies have been proposed, ranging from Partial Differential Equations (PDE)-based pricers to Monte Carlo methods such as the Longstaff & Schwartz pricer [6]. With the emergence of Machine Learning over the past decade, new algorithms have garnered attention. This work aims to explain in details the whys and wherefores of [1], which harness the strengths of Neural Networks to enhance the traditional viewpoint of Stochastic Differential Equations (SDEs). The reader may find all codes used in this project on https://github.com/Noomkwah/NeuralSDE.

Problem overview Let us assume we wish to price a potentially illiquid path-dependent derivative of discounted payoff $\Psi \in L^2(\mathcal{F}_T)$, where we consider the probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P}^{\text{market}})$, for $T > 0$ a fixed maturity, and a constant interest rate $r \in \mathbb{R}$. Under the risk-neutral measure $\mathbb{Q}^{\text{market}}$, this price is $\mathbb{E}^{\mathbb{Q}^{\text{market}}}[\Psi]$. Finding this value will be our goal. To such end, a two-steps procedure will be followed:

- First, we shall design a model for the diffusion of the underlyings $(X_t)_{t\in[0,T]}$, and calibrate it to the data.
- Second, we shall use the previously calibrated model to price the derivative.

In light of our aim to price a potentially highly exotic derivative, the initial crucial step is to develop an accurate modelization of the underlying assets. We will leverage the strength of neural networks to accomplish this task. Let $(W_t)_{t\in[0,T]}$ be a n-dimensional Brownian motion supported on $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{Q})$ so that $\mathbb{Q}$ is the Wiener measure and $\Omega = \mathcal{C}^0([0,T], \mathbb{R}^n)$. We consider the following SDE:

$$\mathrm{d}X_t^\theta = b(t, X_t^\theta, \theta)\,\mathrm{d}t + \sigma(t, X_t^\theta, \theta)\,\mathrm{d}W_t \tag{1}$$

Here, the functions b and σ are in fact the outputs of an overparametrized neural network whose parameters are represented by the vector $\theta \in \Theta$. Thus, we say that (1) is a Neural SDE.
The data used for this calibration corresponds to a set of liquid derivatives

payoffs $\{\Phi_i\}_{i=1}^{N_{\text{prices}}}$ and of their prices $\{\mathfrak{p}(\Phi_i)\}_{i=1}^{N_{\text{prices}}}$. We seek for a model (i.e for a θ) such that under the law $\mathbb{Q}(\theta)$ of $(X_t^\theta)_{t\in[0,T]}$, we have $\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i] = \mathfrak{p}(\Phi_i)$ for $i = 1, \dots, N_{\text{prices}}$. In full generality, given a convex loss function $\ell : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ whose minimum is 0 and is reached (for instance $l(x, y) = |x - y|^2$), we look for model parameters θ^* such that

$$\theta^* \in \arg\min_{\theta\in\Theta} \sum_{i=1}^{N_{\text{prices}}} \ell\big(\,\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i)\big) \tag{2}$$

In practice, this is equivalent to finding θ^* with $\sum \ell\big(\,\mathbb{E}^{\mathbb{Q}(\theta^*)}[\Phi_i], \mathfrak{p}(\Phi_i)\big) = 0$. Due to the inherent overparametrization of the neural network behind b and σ, the space of potential parameters θ is large enough for equation (2) to present infinitely many solutions. As a result, many martingale measures $\mathbb{Q}(\theta)$ may be considered and it makes it hard to know which θ makes $\mathbb{Q}(\theta)$ as close as possible to $\mathbb{Q}^{\text{market}}$. In light of this, instead of directly approximating $\mathbb{E}^{\mathbb{Q}^{\text{market}}}[\Psi]$ by $\mathbb{E}^{\mathbb{Q}^{\theta^*}}[\Psi]$, we will seek for bounds over the arbitrage-free price by solving

$$\begin{aligned}
\theta^{l,*} &\in \arg\min_{\theta\in\Theta} \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi] \quad \text{subject to} \quad \sum_{i=1}^{N_{\text{prices}}} \ell\Big(\,\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i)\Big) = 0 \\
\theta^{u,*} &\in \arg\max_{\theta\in\Theta} \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi] \quad \text{subject to} \quad \sum_{i=1}^{N_{\text{prices}}} \ell\Big(\,\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i)\Big) = 0
\end{aligned} \tag{3}$$

Introductary remarks The reader may notice that the framework of pricing presented here does not rely on data about the historical process $(X_t)_{t\in[0,T]}$. Instead, we directly fit the method on prices of liquid derivatives. This suggests that in the end, the model may come up with a diffusion that significantly differs from the observed reality, while still being accurately calibrated to derivatives data. Moreover, the fit of the diffusion (1) assumes Markovian settings. This hypothesis could be relaxed (by replacing any instance of X_t with the stopped trajectory $(X_{s\wedge t})_{s\in[0,T]}$), but we postpone this relaxation to a future work.

2 Model design and calibration

2.1 Variance reduction

From the problem overview, we know that finding the no-arbitrage price of a given illiquid derivative of payoff Ψ is hazardous, hence the need for bounds $\big[\mathbb{E}^{\mathbb{Q}(\theta^{l,*})}[\Psi], \mathbb{E}^{\mathbb{Q}(\theta^{u,*})}[\Psi]\big]$. In first place, these bounds could be obtained by solving equations (3).

To estimate $\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi]$, consider $(X^{i,\theta})_{i=1}^{N}$ some i.i.d copies of (1) and let $\mathbb{Q}^N(\theta) := \frac{1}{N}\sum_{i=1}^{N}\delta_{X^{i,\theta}}$. By the Central Limit Theorem, we know that

$$\mathbb{P}\left(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi] \in \left[\mathbb{E}^{\mathbb{Q}^N(\theta)}[\Phi] - z_{\alpha/2}\frac{\sigma}{\sqrt{N}}, \mathbb{E}^{\mathbb{Q}^N(\theta)}[\Phi] + z_{\alpha/2}\frac{\sigma}{\sqrt{N}}\right]\right) \to 1 \tag{4}$$

as $N \to \infty$, with $\sigma := \sqrt{\mathbb{V}\mathrm{ar}[\Phi]}$ and $z_{\alpha/2}$ such that

$$1 - \mathbf{CDF}_Z(z_{\alpha/2}) = \alpha/2, \quad Z \sim \mathcal{N}(0,1)$$

To reduce the width of the confidence interval in (4), one could either increase N or find a control variate Φ^{cv} such that

$$\mathbb{E}^{\mathbb{Q}^N(\theta)}[\Phi] = \mathbb{E}^{\mathbb{Q}^N(\theta)}[\Phi^{cv}] \quad \text{and} \quad \mathbb{V}\mathrm{ar}^{\mathbb{Q}^N(\theta)}[\Phi^{cv}] < \mathbb{V}\mathrm{ar}^{\mathbb{Q}^N(\theta)}[\Phi] \tag{5}$$

For Φ such that $\mathbb{E}^{\mathbb{Q}}[|\Phi^2|] < \infty$, the martingale representation theorem states that there exists a unique process Z adapted to $(\mathcal{F}_t)_{t\in[0,T]}$ such that

$$\mathbb{E}^{\mathbb{Q}}\Big[\int_t^T |Z_s|^2\,\mathrm{d}s\Big] < \infty \quad \text{and} \quad \Phi = \mathbb{E}[\Phi \mid \mathcal{F}_0] + \int_0^T Z_s\,\mathrm{d}W_s$$

By defining $\Phi^{cv} := \Phi - \int_0^T Z_s\,\mathrm{d}W_s$, one notices that (5) is verified by Φ^{cv}. Replacing each Φ_i by its alias Φ_i^{cv} therefore appears to be a good idea, as long as we can efficiently compute the Z^i, $i = 1,\dots,N_{\text{prices}}$. Here, the approximation of Z will be provided by another neural network.
Because the data $\{\Phi_i\}_{i=1}^{M}$ corresponds to liquid payoffs over the underlying assets X^θ, we can write without loss of generality:

$$\Phi_i = \phi_i((X_t^\theta)_{t\in[0,T]}) \quad \text{for some} \quad \phi_i : \mathcal{C}^0([0,T],\mathbb{R}) \to \mathbb{R}$$

Let $\mathfrak{h} : [0,T] \times \mathcal{C}^0([0,T],\mathbb{R}^d) \times \mathbb{R}^p \to \mathbb{R}^d$ be a neural network with parameters $\xi \in \mathbb{R}^{p'}$ (whose goal is to approximate Z). We wish $\mathfrak{h}$ to minimize the variance of Φ^{cv}, hence we define the following learning task:

$$\xi^* \in \arg\min_{\xi} \mathbb{V}\mathrm{ar}\left[\phi((X_t^\theta)_{t\in[0,T]}) - \int_0^T \mathfrak{h}(t,(X_{s\wedge t})_{s\in[0,T]},\xi)\,\mathrm{d}W_t \mid \mathcal{F}_0\right] \tag{6}$$

In a similar manner, deriving Ψ^{cv} for the payoff Ψ allows us to restate (3) to:

$$\begin{aligned}\theta^{l,*} &\in \arg\min_{\theta\in\Theta} \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi^{cv}] \quad \text{subject to} \quad \sum_{i=1}^{N_{\text{prices}}} \ell\Big(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i^{cv}], \mathfrak{p}(\Phi_i)\Big) = 0\\ \theta^{u,*} &\in \arg\max_{\theta\in\Theta} \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi^{cv}] \quad \text{subject to} \quad \sum_{i=1}^{N_{\text{prices}}} \ell\Big(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i^{cv}], \mathfrak{p}(\Phi_i)\Big) = 0\end{aligned} \tag{7}$$

2.2 Optimal hedge in incomplete market

The reader may have noticed that training $\mathfrak{h}$ to determine the process Z actually amounts to computing a hedging strategy over the derivative Ψ. However, no assumption were made over X^θ in (1), so that the abstract hedge Z^Ψ may not be usable in practice. Indeed, if we let $X^\theta = (S^\theta, V^\theta)$ were S^θ denotes the tradable assets and V^θ the non-tradable components (we split $\mathfrak{h} = (\mathfrak{h}^S, \mathfrak{h}^V)$ accordingly), then having $\mathfrak{h}^V \neq 0$ corresponds to an impossible strategy. Hence, it is necessary to modify (6). We can rewrite (1) as

$$\begin{aligned}\mathrm{d}S_t^\theta &= rS_t^\theta\,\mathrm{d}t + \sigma^S(t, X_t^\theta, \theta)\,\mathrm{d}W_t\\ \mathrm{d}V_t^\theta &= b^V(t, X_t^\theta, \theta)\,\mathrm{d}t + \sigma^V(t, X_t^\theta, \theta)\,\mathrm{d}W_t\\ X_t^\theta &= (S_t^\theta, V_t^\theta)\end{aligned} \tag{8}$$

where $\sigma = (\sigma^S, \sigma^V)$ and $b = (rs, b^V)$. We are looking for a new $\bar{\mathfrak{h}} := \bar{\mathfrak{h}}^S$ that defines a hedging strategy. Notice that $\bar{S}_t^\theta := e^{-rt}S_t^\theta$ defines a (local) martingale which follows the diffusion

$$\mathrm{d}\bar{S}_t^\theta = e^{-rt}\sigma^S(t, X_t^\theta, \theta)\,\mathrm{d}W_t \quad \Big(\text{and} \quad \mathrm{d}\langle \bar{S}^\theta, \bar{S}^\theta\rangle_t = e^{-2rt}\sigma^S(t, X_t^\theta, \theta)^2\,\mathrm{d}t\Big)$$

Leveraging the martingale representation theorem again, there exists a unique process $\bar{Z}$ adapted to $(\mathcal{F}_t)_{t\in[0,T]}$ such that

$$\mathbb{E}^{\mathbb{Q}}\Big[\int_t^T |\bar{Z}_s|^2\,\mathrm{d}\langle \bar{S}^\theta, \bar{S}^\theta\rangle_s\Big] < \infty \quad \text{and} \quad \Phi = \mathbb{E}[\Phi \mid \mathcal{F}_0] + \int_0^T \bar{Z}_s\,\mathrm{d}\bar{S}_s^\theta$$

Therefore, an alternative approach to (6), possibly yielding a better hedge but worse variance reduction would be to consider finding

$$\bar{\xi}^* \in \arg\min_{\bar{\xi}} \mathbb{V}\mathrm{ar}\left[\phi((X_t^\theta)_{t\in[0,T]}) - \int_0^T \bar{\mathfrak{h}}(t,(X_{s\wedge t})_{s\in[0,T]},\bar{\xi})\,\mathrm{d}\bar{S}_t^\theta \mid \mathcal{F}_0\right] \tag{9}$$

3 Implementation

3.1 Finite schemes

Following the two-steps procedure described in the problem overview, we start by defining the scheme used to approximate the diffusion of (1).
Let π be a partition of $[0,T]$: $\pi := \{0 = t_0 < t_1 < \cdots < t_{N_{\text{steps}}} = T\}$. Depending on the neural network architecture approximating σ and b, we may have super-linear growth in the coefficients of (1), in which case a standard Euler Explicit scheme might not converge. To avoid these issues, we introduce the tamed Euler scheme [3], given by

$$\Delta X_{t_k}^{\pi,\theta} = \frac{b(t_k, X_{t_k}^{\pi,\theta},\theta)}{1+|b(t_k,X_{t_k}^{\pi,\theta},\theta)|\sqrt{\Delta t_k}}\Delta t_k + \frac{\sigma(t_k,X_{t_k}^{\pi,\theta},\theta)}{1+|\sigma(t_k,X_{t_k}^{\pi,\theta},\theta)|\sqrt{\Delta t_k}}\Delta W_{t_k} \tag{10}$$

where

$$\begin{cases} \Delta X_{t_k}^{\pi,\theta} &= X_{t_{k+1}}^{\pi,\theta} - X_{t_k}^{\pi,\theta} \\ \Delta t_k &= t_{k+1} - t_k \\ \Delta W_{t_k} &= W_{t_{k+1}} - W_{t_k} \end{cases}$$

The problem of finding the (bounds for the) price of Ψ as well as its hedge can be summarized through equations (7) and (9). The stochastic integral of (9) is approximated by the appropriate Riemann sum

$$\int_0^T \bar{\mathfrak{h}}(s,X_s^\theta,\bar{\xi})\,\mathrm{d}\bar{S}_s^\theta \quad\longleftrightarrow\quad \sum_{k=0}^{N_{\text{steps}}-1}\bar{\mathfrak{h}}(t_k,(X_{t\wedge t_k})_{t\in\pi},\bar{\xi})\Delta\bar{S}_{t_k}^{\pi,\theta}$$

This leads to the following discretization of (9):

$$\bar{\xi}^* \in \widehat{\arg\min_{\bar{\xi}}} \sum_{j=1}^{N_{\text{prices}}} \mathbb{V}\mathrm{ar}^{\mathbb{Q}^{N_{\text{trn}}(\theta)}}\left[\Phi_j(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{steps}}-1}\bar{\mathfrak{h}}(t_k,X_{t_k},\bar{\xi}_j)\Delta\bar{S}_{t_k}^{\pi,\theta}\right] \tag{11}$$

In (11), we wrote $\bar{\mathfrak{h}}(t_k,X_{t_k},\bar{\xi}_j)$ instead of $\bar{\mathfrak{h}}(t_k,(X_{t\wedge t_k})_{t\in\pi},\bar{\xi}_j)$ because this will speed up the computations. We postpone discretization of (7) in the next section, as it is directly a part of the algorithms involved.

3.2 Algorithm for pricing and hedging

We aim to train neural networks b, σ to calibrate (8) and at the same time find bounds of (7) as well as the hedging strategy $\bar{\mathfrak{h}}$ in (9).

Augmented Lagrangian method The optimization problem (7) consists in two similar constrained objectives. Such a task requires special methods. For the sake of comprehension, we give an outline of the general Augmented Lagrangian method [2]. Let us consider the following constrained optimization problem:

$$\min_{\theta\in\Theta} f(\theta) \quad \text{subject to} \quad h(\theta) = 0 \tag{12}$$

The idea of the Augmented Lagrangian method is to combine regularization with the natural Lagrangian of the problem in an iterative fashion, to ensure better convergence properties. Namely, the following unconstrained problem is solved:

$$\theta_k \in \arg\min_{\theta\in\Theta} f(\theta) + \lambda_k \cdot h(\theta) + c_k/2 \cdot h(\theta)^2 \tag{13}$$

Given arbitrary $\lambda_0, c_0 > 0$, solving (13) yields θ_0. Starting from there, we compute θ_{k+1} by solving (13) again with $\lambda_{k+1} = \lambda_k + c_k \cdot h(\theta_k)$ and $c_{k+1} = 2c_k$, using θ_k as initial guess (the 'warm start'). As $\lambda_k, c_k \to \infty$, the constraint $h(\theta) = 0$ will be more and more accurately satisfied, while $f(\theta)$ will still be minimized. We stop when a convergence threshold is reached.

Back to solving (7), applying the Augmented Lagrangian method is possible, letting

$$\begin{aligned} f(\theta) &:= \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi^{cv}] = \mathbb{E}^{\mathbb{Q}(\theta)}\left[\Psi - \int_0^T \bar{Z}_s^{\Psi}\,\mathrm{d}\bar{S}_s^{\theta}\right] \\ h(\theta) &:= \sum_{i=1}^{N_{\text{prices}}} \ell\left(\mathbb{E}^{\mathbb{Q}(\theta)}\left[\Phi_i - \int_0^T \bar{Z}_s^{i}\,\mathrm{d}\bar{S}_s^{\theta}\right], \mathfrak{p}(\Phi_i)\right) \end{aligned} \tag{14}$$

Algorithm The data at our disposal consists in N_{prices} couples (payoff, price) of liquid derivatives. In practice, N_{prices} may be really low in front of the number of parameters to optimize in θ and ξ. Therefore, for the training to be efficient, many epochs might be necessary. Instantiating θ and ξ with random values, we proceed as follows in each of the N_{epochs} epochs:

- **Optimization of θ, with ξ fixed:** Using the tamed Euler scheme, we generate N_{trn} paths $(x_{t_n}^{\pi,\theta,i})_{n=0}^{N_{\text{steps}}} := (s_{t_n}^{\pi,\theta,i}, v_{t_n}^{\pi,\theta,i})_{n=0}^{N_{\text{steps}}}$, for $i = 1, \dots, N_{\text{trn}}$. To prevent the optimizer from modifying these samples during the backward propagation step, we make use of copies of these paths, denoted by $(\tilde{x}_{t_n}^{\pi,i})_{n=0}^{N_{\text{steps}}}$. These copies do not depend on θ anymore.
  Freezing ξ, we use Adam [4] to update θ by the Augmented Lagrangian method:
$$\theta \in \widehat{\arg\min_{\theta}}\, f(\theta) + \lambda \cdot h(\theta) + c \cdot h(\theta)^2 \tag{15}$$
  where
$$f(\theta) := \mathbb{E}^{\mathbb{Q}^{N_{\text{trn}}(\theta)}} \left[\Psi(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{steps}-1}} \bar{\mathfrak{h}}\big(t_k, \tilde{X}_{t_k}^{\pi}, \xi_\Psi\big) \Delta \tilde{\bar{S}}_{t_k}^{\pi} \right]$$
$$h(\theta) := \sum_{j=1}^{N_{\text{prices}}} \ell \left(\mathbb{E}^{\mathbb{Q}^{N_{\text{trn}}(\theta)}} \left[\Phi_j(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{steps}-1}} \bar{\mathfrak{h}}\big(t_k, \tilde{X}_{t_k}^{\pi}, \xi_j\big) \Delta \tilde{\bar{S}}_{t_k}^{\pi} \right],\ \mathfrak{p}(\Phi_j) \right)$$
  To make up for the potential slowness of convergence of θ (due to a potentially too low learning rate for instance), updates of λ and c are however performed every 50 epochs.

- **Optimization of ξ, with θ fixed:** Again, we use Adam to update $\xi = (\xi_1, \dots, \xi_{N_{\text{prices}}}, \xi_\Psi)$ by solving:
$$\xi \in \widehat{\arg\min_{\xi}}\, V_\Psi + \sum_{j=1}^{N_{\text{prices}}} V_j \tag{16}$$
  where
$$V_j = \mathbb{V}\text{ar}^{\mathbb{Q}^{N_{\text{trn}}(\theta)}} \left[\Phi_j(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{prices}}-1} \bar{\mathfrak{h}}\big(t_k, X_{t_k}^{\pi,\theta}, \xi_j\big) \Delta \tilde{\bar{S}}_{t_k}^{\pi} \right]$$
$$V_\Psi = \mathbb{V}\text{ar}^{\mathbb{Q}^{N_{\text{trn}}(\theta)}} \left[\Psi(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{prices}}-1} \bar{\mathfrak{h}}\big(t_k, (X_{t\wedge t_k}^{\pi,\theta})_{t\in\pi}, \xi_j\big) \Delta \tilde{\bar{S}}_{t_k}^{\pi} \right]$$

Enhancing efficiency In the case where training data consists in call options of different maturities $(T_0 :=)\, 0 < T_1 < \cdots < T_{N_{\text{maturities}}} \leq T$, we let b and σ be the sum of $N_{\text{maturities}}$ distinct neural networks outputs (one for each maturity), so that

$$b(t, X_t, \theta) = \sum_{m=1}^{N_{\text{maturities}}} \mathbf{1}_{t \in [T_{m-1}, T_m)} \cdot b_m(t, X_t, \theta_m)$$
$$\sigma(t, X_t, \theta) = \sum_{m=1}^{N_{\text{maturities}}} \mathbf{1}_{t \in [T_{m-1}, T_m)} \cdot \sigma_m(t, X_t, \theta_m)$$

This way, we do not update the whole vector $\theta := (\theta_1, \ldots, \theta_{N_{\text{maturities}}})$ at each pass but only one part of it.

4 Simulation results

We implemented a simple version of the algorithm and tested it over two models: the local volatility model and the local stochastic volatility model. In both cases, $X = S$ (there is no untradable component V), and our goal was to evaluate the lookback put option, whose payoff at $t = T$ is:

$$\mathrm{LP}((S_t)_{t \in [0,T]}) = S_{\max} - S_T$$

where $S_{\max}$ denote the maximum values S takes over $[0, T]$.

We chose the mean squared error as our loss function ℓ, and decided to calibrate the neural networks on available vanilla prices (minimizing $h(\theta)$ instead of (15)). As we did not have access to real data, we simulated them. The data generation process is described in appendix A.
To ensure the best performances both for the forward and the backward pass of the neural networks, we run the computations on GPU (with CUDA).

4.1 Local Volatility model

The local volatility model assumes that there exists a function σ such that

$$\mathrm{d}S_t = rS_t\,\mathrm{d}t + S_t\sigma(t, S_t)\,\mathrm{d}W_t, \quad S_0 = s_0 > 0 \tag{17}$$

Equation (17) is rewritten in the form

$$\mathrm{d}S_t^\theta = rS_t^\theta \, \mathrm{d}t + S_t^\theta \sigma(t, S_t^\theta, \theta) \, \mathrm{d}W_t, \quad S_0^\theta = s_0 > 0 \tag{18}$$

to abide by the general framework of Neural SDEs. The goal is then to calibrate the neural network σ to vanilla prices.

In the process of training the Neural Local Volatility (LV) model described above (which consists of around 100,000 parameters to optimize), we achieved a mean squared error (MSE) of $1.5 \cdot 10^{-8}$ over the training dataset after 1000 epochs. The model's accuracy is best assessed by examining figure 1, which illustrates the fit of the implied volatility individual curves (for each maturity) estimated by the Neural LV model. This figure provides compelling evidence of the model's high accuracy, particularly for options with longer maturities. The reader must be aware that the apparent miss-calibration that can be seen on the implied volatility curve of maturity 1/6 years (i.e 2 months) only resides in the high instability of the inversion process by which one obtain the implied volatility from the price of the call option. Indeed, at this point, the observed price for the call option of strike 0.8 and maturity 1/6 was 0.2037, and the model predicted 0.2033.
Training two additional models on the Augmented Lagrangian problem provided a lower bound of 0.1601 and an upper bound of 0.1852 for the exotic option price (with 40,000 paths used). It must be highlighted that the standard model did always produce a price in this range.

4.2 Local Stochastic Volatility model

The local stochastic volatility model assumes that there exists functions σ^S, σ^V, b^V such that

$$\begin{aligned} \mathrm{d}S_t &= rS_t \, \mathrm{d}t + S_t \sigma^S(t, S_t, V_t) \, \mathrm{d}W_t^S, && S_0 = s_0 > 0 \\ \mathrm{d}V_t &= b^V(V_t) \, \mathrm{d}t + \sigma^V(V_t) \, \mathrm{d}W_t^V, && V_0 = v_0 > 0 \\ \mathrm{d}\langle W^S, W^V \rangle_t &= \rho \, \mathrm{d}t \end{aligned} \tag{19}$$

Again, denoting $\theta = (\alpha, \beta, \gamma, \rho, v_0)$, we model:

$$\begin{aligned} \mathrm{d}S_t^\theta &= rS_t^\theta \, \mathrm{d}t + S_t^\theta \sigma^S(t, S_t^\theta, V_t^\theta, \alpha) \, \mathrm{d}W_t^S, && S_0^\theta = s_0 > 0 \\ \mathrm{d}V_t^\theta &= b^V(V_t^\theta, \beta) \, \mathrm{d}t + \sigma^V(V_t^\theta, \gamma) \, \mathrm{d}W_t^V, && V_0^\theta = v_0 > 0 \\ \mathrm{d}\langle W^S, W^V \rangle_t &= \rho \, \mathrm{d}t \end{aligned} \tag{20}$$

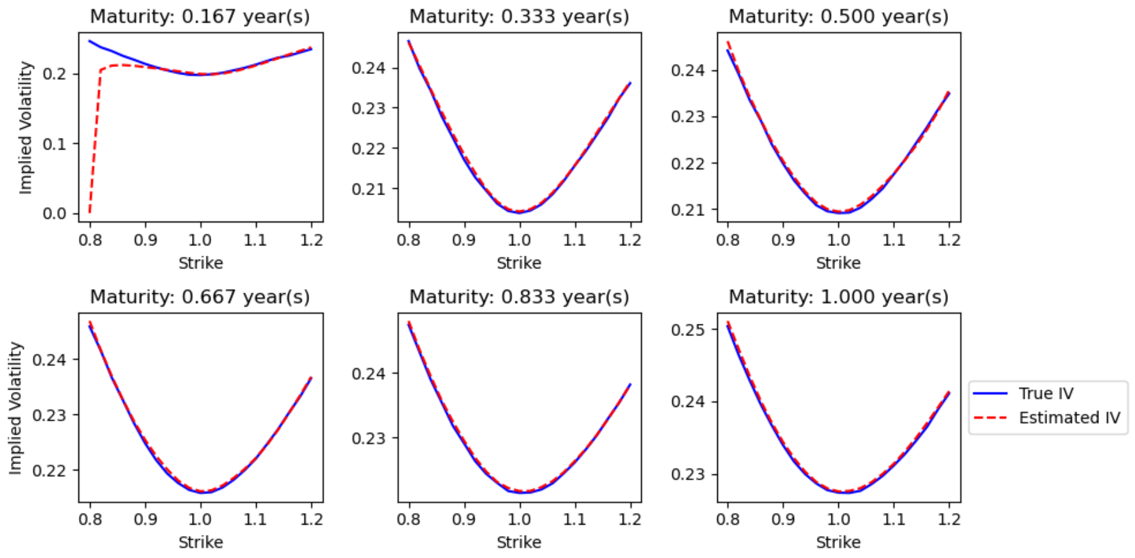

Figure 1: Implied Volatility individual curves for the standard Neural LV model

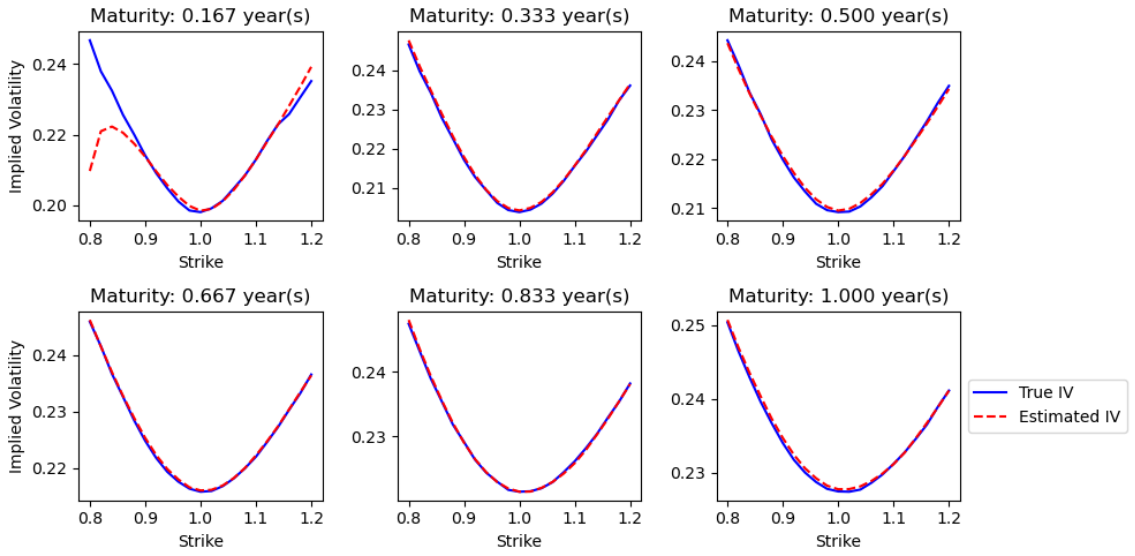

Figure 2: Implied Volatility individual curves for the standard Neural LSV model

A 1000 epochs long training resulted in a model (with around 200,000 parameters) which achieved an MSE of $4 \cdot 10^{-9}$. Figure 2 displays the estimated implied volatility derived from the model. The lower and upper bounds for the exotic option price for this model were 0.1671 and 0.1896, respectively. Similarly, the standard model consistently produced prices within this range, approximately 0.1723.
As each training session required 5 to 13 hours of computation, we lacked the computational resources to conduct more than one simulation for each model, despite the potential benefits of running multiple simulations.

5 Conclusion and perspectives

In this work, we reviewed a new pricing framework: Neural SDEs. By leveraging Neural Networks, we were able to calibrate an SDE to a few market data points with high accuracy and robustly price exotic derivatives such as the Lookback put option. Moreover, Neural SDEs demonstrated strong precision over individual implied volatility surfaces. With a correctly trained Neural SDE, pricing illiquid derivatives becomes a task that can be performed quickly, meeting industry needs.
However, it should be emphasized that this approach presents two drawbacks: first, the output's variance is at the moment too high for this model to be reliable; second, the fact that the control variate for illiquid derivatives is learned during training for the lower and upper bounds restricts the use of such a network to pre-selected illiquid derivatives.
Nevertheless, the relevance of this work lies in its unparalleled computational speed as well as its great adaptability. Once the network is trained, the issue of interpretability and robustness of the outputs has been partially addressed by the authors through the computation of lower and upper bounds.

References

[1] Patryk Gierjatowicz, Marc Sabate-Vidales, David Šiška, Lukasz Szpruch, and Žan Žurič. Robust pricing and hedging via neural sdes, 2020.

[2] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969.

[3] Huimin Hu and Siqing Gan. Strong convergence of the tamed euler scheme for scalar sdes with superlinearly growing and discontinuous drift coefficient, 2022.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[5] Steven L. and 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.

[6] Francis Longstaff and Eduardo Schwartz. Valuing american options by simulation: A simple least-squares approach. *Review of Financial Studies*, 14:113–47, 02 2001.

A Data used in calibration

The data used in order to calibrate the model were prices of calls and puts, generated using a Heston model [5] given by

$$\begin{aligned} \mathrm{d}X_t &= rX_t\,\mathrm{d}t + X_t\sqrt{V_t}\,\mathrm{d}W_t, & X_0 = x_0 \\ \mathrm{d}V_t &= \kappa(\mu - V_t)\,\mathrm{d}t + \eta\sqrt{V_t}\,\mathrm{d}B_t, & V_0 = v_0 \\ \mathrm{d}\langle B, W\rangle_t &= \rho\,\mathrm{d}t \end{aligned}$$

where B and W are two brownian motions adapted to $(\mathcal{F}_t)_{t\in[0,T]}$.
The choice of parameters is the following:

$$x_0 = 1,\ \ r = 0.025,\ \ \kappa = 0.78,\ \ \mu = 0.11,\ \ \eta = 0.68,\ \ V_0 = 0.04,\ \ \rho = 0.044$$

Under the Heston model, a semi-closed formula for the price at t of a call of maturity T and strike K does exist [5], but it is well-known that this formula is numerically unstable for extreme values of K and T. Therefore, we implemented the Euler scheme given by

$$\begin{aligned} \Delta X^{\pi}_{t_k} &= rX^{\pi}_{t_k}\Delta t_k + X^{\pi}_{t_k}\sqrt{V^{\pi,+}_{t_k}}\Delta W^{\pi}_{t_k} \\ \Delta V^{\pi,+}_{t_k} &= \kappa(\mu - V^{\pi,+}_{t_k})\Delta t_k + \eta\sqrt{V^{\pi,+}_{t_k}}\left(\rho\Delta W^{\pi}_{t_k} + \sqrt{1-\rho^2}\Delta\tilde{W}^{\pi}_{t_k}\right) \end{aligned}$$

where π is a partition of $[0,T]$, W and $\tilde{W}$ are two independent brownian motions, $x^+ := \max(0,x)$, and $\Delta A_{t_k} := A_{t_{k+1}} - A_{t_k}$.
We computed the prices of call options for six maturities ranging from 2 monts to 1 years and for a range of 21 strikes between 0.8 and 1.2. Figure 3 displays the implied volatility surface associated with these data.

B Various plots

Figure 4 depicts the evolution of the MSE during the training of the Neural LV model, as well as the evolution of the hedging loss (defined as $V_\Psi + \sum_{j=1}^{N_{\text{prices}}} V_j$ in (16)).

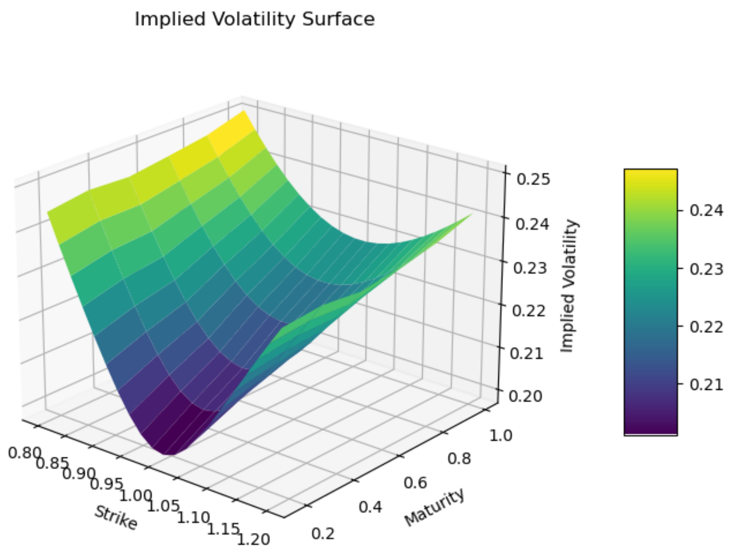

Figure 3: Implied Volatility Surface of the generated data

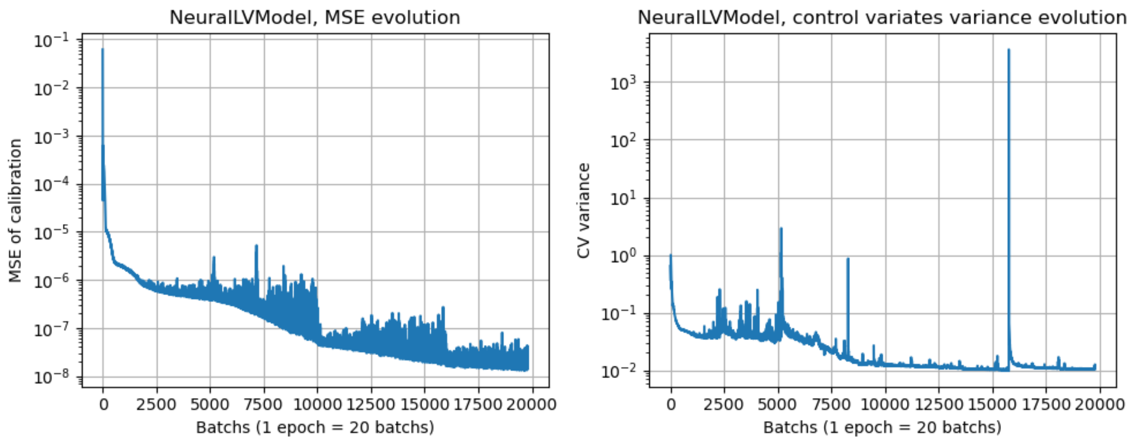

Figure 4: MSE of calibration to market data and cumulated variance during training for the Neural LV model

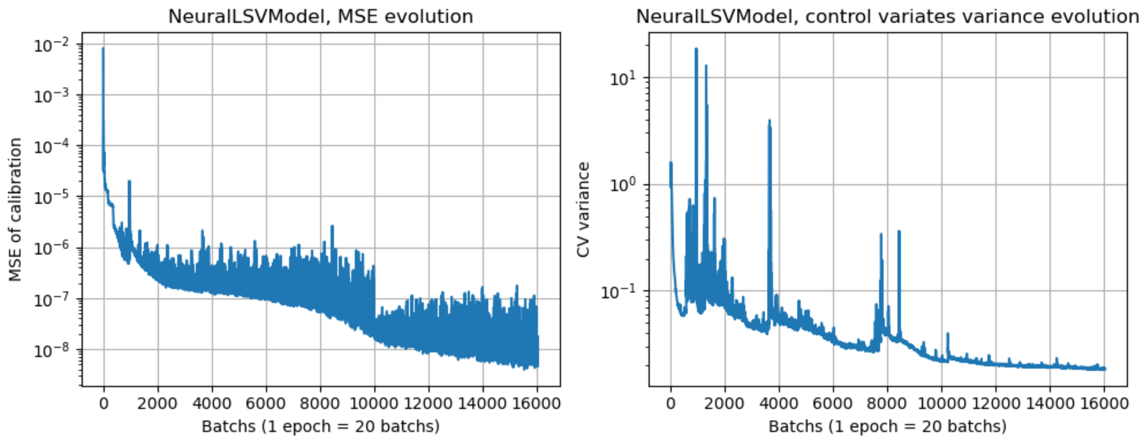

Figure 5: MSE of calibration to market data and cumulated variance during training for the Neural LSV model