Robust Pricing and Hedging via Neural SDEs: a tutorial

Keryann MASSIN & Anna LIFFRAN

ENSAE Paris

Presentation Overview

Introduction - Problem Overview

Objective: Pricing an illiquid derivative of discounted payoff $\Psi \in L^2(\mathcal{F}_T)$ (interest rate $r > 0$ is given and fixed).

Two-step process: Calibrate the diffusion of the underlying $(X_t)_{t\in[0,T]}$ and price Ψ.

How? By using neural networks! We talk about *Neural SDEs*.

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

Introduction - Data and training

Data: Observed prices of liquid derivatives (call options for instance). We denote their payoffs by $\{\Phi_i\}_{i=1}^{N_{\text{prices}}}$ and their prices by $\{\mathfrak{p}(\Phi_i)\}_{i=1}^{N_{\text{prices}}}$.

Optimization problem: To calibrate the neural networks b and σ, we must find θ^* such that

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

where $\mathbb{Q}(\theta)$ denotes the law of $(X_t^\theta)_{t\in[0,T]}$, and ℓ is a loss (e.g. $\ell(x,y) = |x-y|^2$).

Introduction - Robust bounds

Arising issue: (2) may present many solutions. Thus, $\mathbb{Q}^{\theta^*}$ might not be similar to $\mathbb{Q}^{\text{market}}$.

Potential solution: Instead of solving (2), we can provide lower and upper bounds for the output price of the illiquid derivative Ψ. We seek for these bounds by solving

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

Model design and calibration - Variance reduction

Issue: To compute $\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi]$, we need to simulate many Monte Carlo paths, which is memory and resources expensive.

Solution: Leveraging the *Martingale representation theorem*, we look for $(Z_t)_{t\in[0,T]}$ such that $\Phi^{cv} := \Phi - \int_0^T Z_s \,\mathrm{d}W_s$ is a control variate.

Second learning task: We shall approximate Z_t by a new neural network $\mathfrak{h}$, of parameters ξ, whose goal is to solve

$$\xi^* \in \arg\min_{\xi} \mathbb{V}\mathrm{ar}\left[\phi((X_t^\theta)_{t\in[0,T]}) - \int_0^T \mathfrak{h}(t, (X_{s\wedge t})_{s\in[0,T]}, \xi)\,\mathrm{d}W_t \mid \mathcal{F}_0\right] \tag{4}$$

In (4), we then replace any Φ by Φ^{cv}.

Model design and calibration - Optimal hedge

Issue: In case where $X = (S, V)$ with a non-tradable component V, the abstract hedge Z found is not usable in practice, as it relies on exchanges of V.

Solution: We must alter the hedging strategy optimization problem in practice, from (4) to (5):

$$\bar{\xi}^* \in \arg\min_{\bar{\xi}} \mathbb{V}\mathrm{ar}\left[\phi((X_t^\theta)_{t\in[0,T]}) - \int_0^T \bar{\mathfrak{h}}(t,(X_{s\wedge t})_{s\in[0,T]},\bar{\xi})\,\mathrm{d}\bar{S}_t^\theta \mid \mathcal{F}_0\right] \tag{5}$$

where $\mathrm{d}S_t^\theta = rS_t^\theta\,\mathrm{d}t + \sigma^S(t, X_t^\theta, \theta)\,\mathrm{d}W_t$ and $\bar{S}_t^\theta := e^{-rt}S_t^\theta$ defines a local martingale.

Implementation - Finite schemes

Tamed Euler scheme: To diffuse X^θ from $t = 0$ to $t = T$, we define a time mesh $\pi := \{0 = t_0 < t_1 < \cdots < t_{N_{\text{steps}}} = T\}$ and rely on the Tamed Euler scheme.

$$\Delta X_{t_k}^{\pi,\theta} = \frac{b(t_k, X_{t_k}^{\pi,\theta}, \theta)}{1 + |b(t_k, X_{t_k}^{\pi,\theta}, \theta)|\sqrt{\Delta t_k}} \Delta t_k + \frac{\sigma(t_k, X_{t_k}^{\pi,\theta}, \theta)}{1 + |\sigma(t_k, X_{t_k}^{\pi,\theta}, \theta)|\sqrt{\Delta t_k}} \Delta W_{t_k} \tag{6}$$

Discretization of the hedge:

$$\bar{\xi}^* \in \widehat{\arg\min_{\bar{\xi}}} \sum_{j=1}^{N_{\text{prices}}} \mathbb{V}\text{ar}^{\mathbb{Q}^{N_{\text{trn}}}(\theta)} \left[\Phi_j(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{steps}}-1} \bar{\mathfrak{h}}(t_k, X_{t_k}, \bar{\xi}_j) \Delta \bar{S}_{t_k}^{\pi,\theta} \right] \tag{7}$$

Implementation - Algorithm for pricing and hedging

Augmented Lagrangian: To solve (3), we iteratively solve the following optimization problems

$$\theta_k \in \underset{\theta \in \Theta}{\arg\min} \; \pm f(\theta) + \lambda_k \cdot \mathrm{MSE}(\theta) + c_k/2 \cdot \mathrm{MSE}(\theta)^2 \tag{8}$$

where $\lambda_{k+1} = \lambda_k + c_k \mathrm{MSE}(\theta)$ and $c_{k+1} = 2c_k$ $(\lambda_0, c_0 > 0)$. $f(\theta)$ is the price of the exotic derivative.

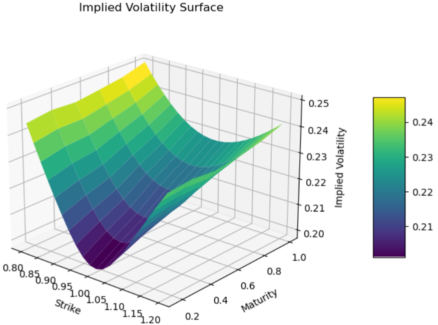
In practice: We started with $\lambda_0 = 10,000$ and $c_0 = 20,000$ and bounded $\lambda_k < 10^6, c_k < 10^{10}$.

Simulation results - Data

Exotic product chosen: Lookback put option, of payoff

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

Data to train models: 121 call options with strikes in $[0.8, 1.2]$ and maturities in $[1/6, 1]$. Data generated using 10^7 MC paths of an Heston model.

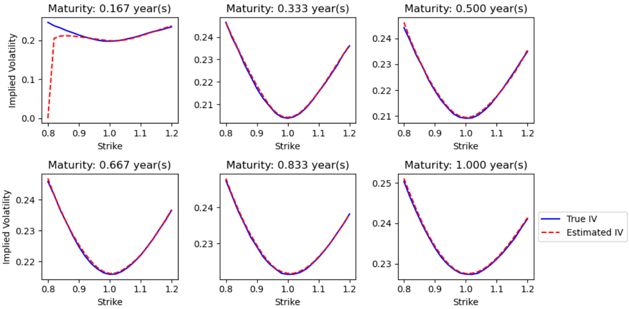


Simulation results - Local Volatility model

Diffusion:

$$\mathrm{d}S_t^\theta = rS_t^\theta\,\mathrm{d}t + S_t^\theta\sigma(t, S_t^\theta, \theta)\,\mathrm{d}W_t, \quad S_0^\theta = s_0 > 0 \tag{9}$$

Calibration error: We estimated the Implied Volatity curves below.

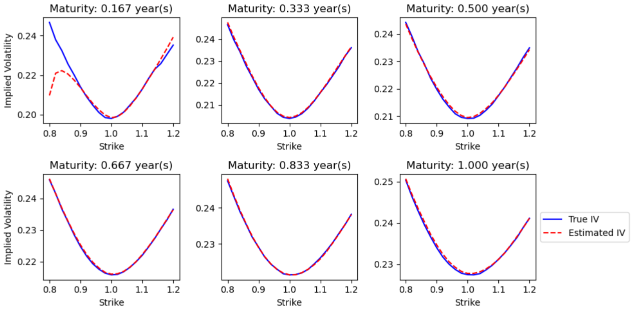


Simulation results - Local Stochastic Volatility model

Diffusion:

$$
\begin{aligned}
\mathrm{d}S_t^\theta &= rS_t^\theta\,\mathrm{d}t + S_t^\theta \sigma^S(t, S_t^\theta, V_t^\theta, \alpha)\,\mathrm{d}W_t^S, & S_0^\theta &= s_0 > 0 \\
\mathrm{d}V_t^\theta &= b^V(V_t^\theta, \beta)\,\mathrm{d}t + \sigma^V(V_t^\theta, \gamma)\,\mathrm{d}W_t^V, & V_0^\theta &= v_0 > 0 \quad (10)
\end{aligned}
$$

$$
\mathrm{d}\langle W^S, W^V\rangle_t = \rho\,\mathrm{d}t
$$



Simulation results - Exotic pricing

Price of the Exotic Lookback Put			
Model	Problem	Price	MSE on call prices
LV	lower bound	0.1601	$5 \cdot 10^{-8}$
LV	standard	0.1813	$1 \cdot 10^{-8}$
LV	upper bound	0.1852	$2 \cdot 10^{-7}$
LSV	lower bound	0.1671	$1 \cdot 10^{-8}$
LSV	standard	0.1723	$4 \cdot 10^{-9}$
LSV	upper bound	0.1896	$7 \cdot 10^{-7}$

Conclusion & Perspectives

New framework: Neural SDEs, highly accurate calibration is possible, and computation speed is excellent.

Drawbacks: Output's variance not low enough, and difficulties to correctly train the model on computing lower/upper bounds. Also, usage restricted to predefinite exotic payoffs.

Perspectives: Include path-dependent volatility and drift.

Thanks for your attention