Neural joint S&P 500/VIX smile calibration

Julien Guyon

Ecole des Ponts ParisTech

Joint work with Scander Mustapha
PACM, Princeton University

Financial Engineering Workshop
Bayes Business School, January 31, 2024

julien.guyon@enpc.fr

Take-away message

- We calibrate neural stochastic differential equations (SDEs) jointly to S&P 500 (SPX) smiles, VIX futures, and VIX smiles.
- Drifts and volatilities are modeled as neural networks.
- Minimizing a suitable loss allows us to fit market data for multiple SPX and VIX maturities.
- A **one-factor Markovian stochastic local volatility (SLV) model** is shown to fit both smiles and VIX futures within bid-ask spreads.
- The joint calibration actually makes it a **pure path-dependent volatility (PDV) model**, confirming the findings in [*The VIX Future in Bergomi Models: Fast Approximation Formulas and Joint Calibration with S&P 500 Skew*, JG '22].

Motivation

- Joint SPX/VIX: important and difficult problem, especially for short maturities
- "Holy Grail of volatility modeling"
- **Parametric models** have produced **approximate fits** (Baldeaux Badran, Bardgett Gourier Leippold, Cont Kokholm, Fouque Saporito, Guyon Lekeufack, Kokholm Stisen, Pacati Pompa Renò, Papanicolaou Sircar, Gatheral Jusselin Rosenbaum, Rosenbaum Zhang, etc.)
- **First exact solution**: the **nonparametric** discrete-time model of JG '20.
  - Minimum-entropy calibration technique $\longleftrightarrow$ nonlinear optimal transport approach
  - Later extended to continuous time to produce exactly and jointly calibrating nonparametric diffusive models (Guo Loeper Obloj Wang, JG '21)

Motivation

- In this work, we also consider **overparametrized diffusive models** to solve the joint calibration problem, using **neural SDEs** (Tzen Raginsky, Jia Benson, Hodgkinson et al, Gierjatowicz et al, Cuchiero et al, Kidger et al, etc.).
- **Drift and diffusion coefficients are modeled as neural networks**
- Potentially many more parameters than options and futures to calibrate to.
- By the universal approximation theorem for neural networks, neural SDEs have the potential of approximating any SDE.
- Minimizing a loss function over the many parameters, we build a diffusive model that solves the joint calibration problem **within bid-ask spreads**.
- Here we only consider **two-dimensional Markovian neural SDEs**.
- $X = \log\text{-spot}$, Y drives the instantaneous volatility together with X.
- Main benefit of overparametrization: it offers a lot of flexibility
- Main drawback: lack of interpretability of parameters and Y

Comparison with other nonparametric/overparametrized SDE approaches

- Guo et al use an $(n+1)$-dimensional SDE, where n is the number of calibrated VIX expiries.
- The Schrödinger bridge approach of JG '21 requires fine-tuning the vol-of-vol coefficient, which is not optimized upon.
- **Our main contribution**: we show that **the joint calibration problem can be solved with a 2-dimensional Markovian SDE**, whatever the number of calibrated VIX expiries, without bothering about choosing the vol-of-vol.
- Similar in spirit to Abi Jaber Illand Li, who optimize a 2-dimensional SDE (with Y an O-U process) over 6-8 parameters and an input variance curve.
- More generally Cuchiero et al model the volatility as a linear function of the (time extended) signature of a primary polynomial diffusion.
- Interestingly, the joint calibration actually forces the SLV model to be a **pure path-dependent volatility (PDV) model**, confirming the findings in JG '22: the "spot-vol" correlation is pushed to its lower bound -1.
- PDV models have recently been shown to be good candidates for approximately solving the joint calibration problem (Gatheral Jusselin Rosenbaum, Guyon Lekeufack).

The model (M)

$$\begin{cases} dX_t = -\frac{1}{2}\sigma_X(t, X_t, Y_t)^2 dt + \sigma_X(t, X_t, Y_t) dB_t^1, \\ dY_t = \mu_Y(t, X_t, Y_t) dt + \sigma_Y(t, X_t, Y_t) \left(\rho(t, X_t, Y_t) dB_t^1 + \sqrt{1 - \rho(t, X_t, Y_t)^2} dB_t^2 \right) \end{cases}$$

- **General one-factor Markovian SLV model**
- $f_{t,u} := S_t \exp\left(\int_t^u (r_s - q_s) ds\right)$ is the SPX u-forward at time t
- $X_t := \log S_t / f_{0,t}$
- $(B_t^1)_{t \geq 0}$ and $(B_t^2)_{t \geq 0}$ are two independent Brownian motions
- $X_0 = Y_0 = 0$

Objective: build σ_X, μ_Y, σ_Y and ρ to jointly calibrate the SPX smiles at $(T_j^s)_{1 \leq j \leq N_T^s}$ and the VIX futures and the VIX smiles at $(T_j^v)_{1 \leq j \leq N_T^v}$.

The VIX in Model (M)

- In this Markov model ($\tau = 30$ days),

$$\mathrm{VIX}_t^2 := -\frac{2}{\tau}\mathbb{E}\left[\log\frac{S_{t+\tau}}{f_{t,t+\tau}}\middle|\mathcal{F}_t\right] = -\frac{2}{\tau}\mathbb{E}\left[X_{t+\tau} - X_t\middle|\mathcal{F}_t\right] = \mathbb{E}\left[R_t|X_t, Y_t\right] =: v(t, X_t, Y_t)$$

where R_t is the realized variance over 30 days,

$$R_t := \frac{1}{\tau}\int_t^{t+\tau} \sigma_X(s, X_s, Y_s)^2 ds \tag{1}$$

- The VIX is defined by $\mathrm{VIX}_t = \sqrt{\mathrm{VIX}_t^2}$

Neural calibration

We look for $\sigma_X, \mu_Y, \sigma_Y, \rho \in \operatorname{argmin} L(\sigma_X, \mu_Y, \sigma_Y, \rho)$ where the loss L is defined by

$$L(\sigma_X, \mu_Y, \sigma_Y, \rho) = w_{f\mathrm{VIX}} \frac{1}{N_T^v} \sum_{j=1}^{N_T^v} \left(\frac{f\mathrm{VIX}_m(T_j^v)}{f\mathrm{VIX}(T_j^v)} - 1 \right)^2$$

$$+ w_{\mathrm{SPX}} \frac{1}{N_T^s} \sum_{j=1}^{N_T^s} \frac{1}{|D_j^{\mathrm{SPX}|}} \sum_{k \in K_j^{\mathrm{SPX}}} \Delta^{\mathrm{SPX}}(T_j^s, k) \left(\frac{I_m^{\mathrm{SPX}}(T_j^s, k)}{I^{\mathrm{SPX}}(T_j^s, k)} - 1 \right)^2$$

$$+ w_{\mathrm{VIX}} \frac{1}{N_T^v} \sum_{j=1}^{N_T^v} \frac{1}{|D_j^{\mathrm{VIX}|}} \sum_{k \in K_j^{\mathrm{VIX}},\ k > f\mathrm{VIX}_m(T_j^v)} \Delta^{\mathrm{VIX}}(T_j^v, k) \left(\frac{C_m^{\mathrm{VIX}}(T_j^v, k)}{C^{\mathrm{VIX}}(T_j^v, k)} - 1 \right)^2$$

$$+ w_{\mathrm{VIX}} \frac{1}{N_T^v} \sum_{j=1}^{N_T^v} \frac{1}{|D_j^{\mathrm{VIX}|}} \sum_{k \in K_j^{\mathrm{VIX}},\ k \leq f\mathrm{VIX}_m(T_j^v)} \Delta^{\mathrm{VIX}}(T_j^v, k) \left(\frac{P_m^{\mathrm{VIX}}(T_j^v, k)}{P^{\mathrm{VIX}}(T_j^v, k)} - 1 \right)^2$$

Neural calibration

- Positive weights $w = (w_{f\mathrm{VIX}}, w_{\mathrm{SPX}}, w_{\mathrm{VIX}})$
- Small bid-ask spreads are given more importance:

$$\Delta^{\mathrm{SPX}}(T_j^s, k) = \frac{\delta^{\mathrm{SPX}}(t,k)}{\sum_{l \in K_j^{\mathrm{SPX}}} \delta^{\mathrm{SPX}}(t,l)}, \quad \Delta^{\mathrm{VIX}}(T_j^v, k) = \frac{\delta^{\mathrm{VIX}}(t,k)}{\sum_{l \in K_j^{\mathrm{VIX}}} \delta^{\mathrm{VIX}}(t,l)}$$

  where $\delta^{\mathrm{SPX}}(t,k)$ (resp. $\delta^{\mathrm{VIX}}(t,k)$) denotes the inverse of the bid-ask spread of the OTM implied volatility $I^{\mathrm{SPX}}(t,k)$ (resp. $I^{\mathrm{VIX}}(t,k)$)
- We calibrate the VIX call/put prices instead of the VIX implied volatilities

Loss approximation

- Time discretization (Euler-Maruyama) at dates t_n
- N Monte Carlo paths:

$$\widehat{C}_m^{\mathrm{SPX}}(t_n,k) := \frac{1}{N}\sum_{i=1}^{N}(S_{t_n}^i - k)_+ \longrightarrow \widehat{I}_m^{\mathrm{SPX}}(t_n,k)$$

- $\mathrm{VIX}_{t_n,i}^2 := \mathrm{VIX}^2(t_n, X_{t_n}^i, Y_{t_n}^i) := \mathbb{E}\left[R_{t_n}^i \middle| X_{t_n}^i, Y_{t_n}^i\right]$ where

$$R_{t_n}^i = \frac{\Delta t}{\tau}\sum_{t_n \le t_m < t_n+\tau} \sigma_X(t_m, X_{t_m}^i, Y_{t_m}^i)^2$$

For $\widehat{\mathrm{VIX}^2}_{t_n,i}$ an estimator of $\mathrm{VIX}_{t_n,i}^2$, denote $\widehat{\mathrm{VIX}}_{t_n,i} := \sqrt{\widehat{\mathrm{VIX}^2}_{t_n,i}}$.

$$\widehat{C}_m^{\mathrm{VIX}}(t_n,k) := \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{\mathrm{VIX}}_{t_n,i} - k\right)_+, \quad \widehat{f\mathrm{VIX}}_m(t_n) := \frac{1}{N}\sum_{i=1}^{N}\widehat{\mathrm{VIX}}_{t_n,i}$$

- $L \to \hat{L}$

Neural parametrization

- We parametrize $(\sigma_X, \mu_Y, \sigma_Y, \rho)$ as neural networks
- Let $\Phi_\theta : \mathbb{R}^3 \to \mathbb{R}^4$ be a neural network with r hidden layers of size l and weights θ:

$$\Phi_\theta : \begin{bmatrix} t \\ x \\ y \end{bmatrix} \in \mathbb{R}^3 \longmapsto \begin{bmatrix} \Phi_\theta^1(t,x,y) \\ \Phi_\theta^2(t,x,y) \\ \Phi_\theta^3(t,x,y) \\ \Phi_\theta^4(t,x,y) \end{bmatrix} \in \mathbb{R}^4.$$

- The activation functions for the hidden layers are hyperbolic tangents. No activation function for the output layer.
- We choose

$$\begin{aligned} \sigma_X &= 1 + \tanh(\Phi_\theta^1) \in (0,2) \\ \sigma_Y &= 1 + \tanh(\Phi_\theta^2) \in (0,2) \\ \mu_Y &= \Phi_\theta^3 \\ \rho &= \tanh(\Phi_\theta^4) \in (-1,1) \end{aligned}$$

Minimization of $\widehat{L}$

- In order to apply a gradient descent algorithm, we need to compute the gradients $\partial_\theta \widehat{L}$.

$$\partial_\theta \widehat{I}_m^{\mathrm{SPX}}(t,k) = \partial_c \widehat{I}_m^{\mathrm{SPX}}(t,k) \left[\frac{1}{N} \sum_{i=1}^N \partial_\theta X_t^i S_t^i 1_{S_t^i \geq k} \right]$$

$$\partial_\theta \widehat{C}_m^{\mathrm{VIX}}(t,k) = \frac{1}{N} \sum_{i=1}^N \partial_\theta \widehat{\mathrm{VIX}}_{t,i} 1_{\widehat{\mathrm{VIX}}_{t,i} \geq k} \qquad \text{(same for VIX puts)}$$

$$\partial_\theta \widehat{f\mathrm{VIX}}_m(t) = \frac{1}{N} \sum_{i=1}^N \partial_\theta \widehat{\mathrm{VIX}}_{t,i}$$

- We must compute the gradients $\partial_c \widehat{I}_m^{\mathrm{SPX}}(t,k)$, $\partial_\theta X_t^i$, and $\partial_\theta \widehat{\mathrm{VIX}}_{t,i}$.
- $\partial_c I_m^{\mathrm{SPX}}(t,k)$ is computed by using the inverse function rule.
- We use backpropagation to compute $\partial_\theta X_t^i$.
- Computation of $\widehat{\mathrm{VIX}}_{t,i}$ and the gradients $\partial_\theta \widehat{\mathrm{VIX}}_{t,i}$?

Differentiable VIX^2 estimator, $\mathrm{VIX}^2_{t,i} = \mathbb{E}\left[R^i_t | X^i_t, Y^i_t\right]$

- **Kernel regression.** For $W : \mathbb{R} \to \mathbb{R}_+$ a kernel and $h_X, h_Y > 0$ two bandwidths, a VIX^2 estimator is given by

$$\widehat{\mathrm{VIX}^2}_{t,i} = \frac{\sum_{j=1}^N R^j_t W_{ij}}{\sum_{j=1}^N W_{ij}}, \qquad W_{ij} := W((X^i_t - X^j_t)^2/h_X + (Y^i_t - Y^j_t)^2/h_Y)).$$

  Optimal bandwidths are typically chosen by cross-validation to ensure the best tradeoff between bias and variance.

- **Linear least squares.** Let $\mathbb{R}_d[X, Y] = \{[X^k Y^l]_{0 \le k+l \le d} \cdot \alpha : \alpha \in \mathbb{R}^m\}$ for $d \in \mathbb{N}$, where $[X^k Y^l]_{0 \le k+l \le d} \cdot \alpha := \sum_{k+l \le d} \alpha_{k,l} X^k Y^l$.

$$\widehat{\mathrm{VIX}^2}_{t,i} = [(X^i_t)^k (Y^i_t)^l]_{0 \le k+l \le d} \cdot \alpha^*,$$

  where

$$\alpha^* \in \mathrm{argmin}_{\alpha \in \mathbb{R}^m} \frac{1}{N} \sum_{i=1}^N \left(R^i_t - [(X^i_t)^k (Y^i_t)^l] \cdot \alpha\right)^2$$

  solves the normal equation $A^T A \alpha = A^T R$ where $R = (R^i_t)_{1 \le i \le N}$ and

$$A = [(X^i_t)^k (Y^i_t)^l]_{1 \le i \le N, 0 \le k+l \le d} \in \mathbb{R}^{N \times m}.$$

Differentiable VIX^2 estimator

- **Nested Monte Carlo.** Let $M \in \mathbb{N}^*$. The nested Monte Carlo estimator is

$$\widehat{\mathrm{VIX}^2}_{t,i} = \frac{1}{M} \sum_{j=1}^{M} R_t^j(X_t^i, Y_t^i)$$

where $(R_t^j(x,y))_{1 \le j \le M}$ are M independent samples of the realized variance given that $(X_t, Y_t) = (x, y)$. Computationally intensive.

- **Partial differential equation.**

$$\mathrm{VIX}_t^2(x,y) = -\frac{2}{\tau} \left(\mathbb{E}\left[X_{t+\tau} | X_t = x, Y_t = y \right] - x \right).$$

$\mathbb{E}\left[X_{t+\tau} | X_t = x, Y_t = y \right]$ solves the PDE

$$\begin{cases} \partial_t u + \frac{1}{2}\sigma_X^2 \partial_{xx}^2 u + \frac{1}{2}\sigma_Y^2 \partial_{yy}^2 u + 2\rho\sigma_X\sigma_Y \partial_{xy}^2 u - \frac{1}{2}\sigma_X^2 \partial_x u + \mu_Y \partial_y u = 0 \\ u(t+\tau, x, y) = x \end{cases}$$

Can be solved numerically by alternating direction implicit (ADI) method for example. Accurate but computationally intensive and requires the crucial selection of the discretization grid and boundary conditions.

Differentiable VIX^2 estimator

- We find that a good compromise between accuracy and memory usage is to choose the **least squares method**.
- Once our model has been calibrated, in order to **check the accuracy** of the least squares method, we compare the VIX and VIX smile obtained with the least squares method with those estimated with the (slow) **nested Monte Carlo algorithm**.

Algorithms

Algorithm Differentiable VIX least squares estimator

function COMPUTEVIX($X_t, \partial_\theta X_t, Y_t, \partial_\theta Y_t, R_t, \partial_\theta R_t, d$)

$A \leftarrow [X_t^{ik} Y_t^{il}]_{1 \leq i \leq N, 0 \leq k+l \leq d}$

$Q, \partial_A Q, S, \partial_A S \leftarrow$ DECOMPOSITIONQR(A)

$\alpha^*, \partial_S \alpha^*, \partial_{Q^T R} \alpha^* \leftarrow$ SOLVETRIANGULAR$(S, Q^T R_t)$

$\partial_\theta \alpha^* \leftarrow \partial_\theta A \left[\partial_A S \partial_S \alpha^* + \partial_A Q^T R_t \partial_{Q^T R} \alpha^* \right]$

$\mathrm{VIX}^2 \leftarrow \alpha^* \cdot A$

$\partial_\theta \mathrm{VIX}^2 \leftarrow \alpha^* \partial_\theta A + \partial_\theta \alpha^* A$

$\mathrm{VIX}, \partial_\theta \mathrm{VIX} \leftarrow \sqrt{\mathrm{VIX}^2}, \frac{\partial_\theta \mathrm{VIX}^2}{2\sqrt{\mathrm{VIX}^2}}$

return $\mathrm{VIX}, \partial_\theta \mathrm{VIX}$

end function

Algorithms

Algorithm Training step

function $\textsc{Train}(\theta, N, \Delta t, w, d, lr, \Delta B^1, \Delta B^2)$

$(X_{t_n}, \partial_\theta X_{t_n}, Y_{t_n}, \partial_\theta Y_{t_n}, R_{t_n}, \partial_\theta R_{t_n})_{0 \le t_n \le T} \leftarrow \texttt{Euler}(\Delta t, T, 0, 0, \theta, \Delta B_1, \Delta B_2)$.

for $j \leftarrow 1$ to N_T^v **do**

$\widehat{\mathrm{VIX}}_{T_j^v}, \partial_\theta \widehat{\mathrm{VIX}}_{T_j^v} \leftarrow \textsc{ComputeVIX}(X_{T_j^v}, \partial_\theta X_{T_j^v}, Y_{T_j^v}, \partial_\theta Y_{T_j^v}, R_{T_j^v}, \partial_\theta R_{T_j^v}, d)$

end for

$L, \partial_\theta L \leftarrow \textsc{ComputeL}(T, (X_{t_n}, \partial_\theta X_{t_n})_{0 \le t_n \le T}, (\widehat{\mathrm{VIX}}_{T_j^v}, \partial_\theta \widehat{\mathrm{VIX}}_{T_j^v})_{1 \le j \le N_T^v}, w)$.

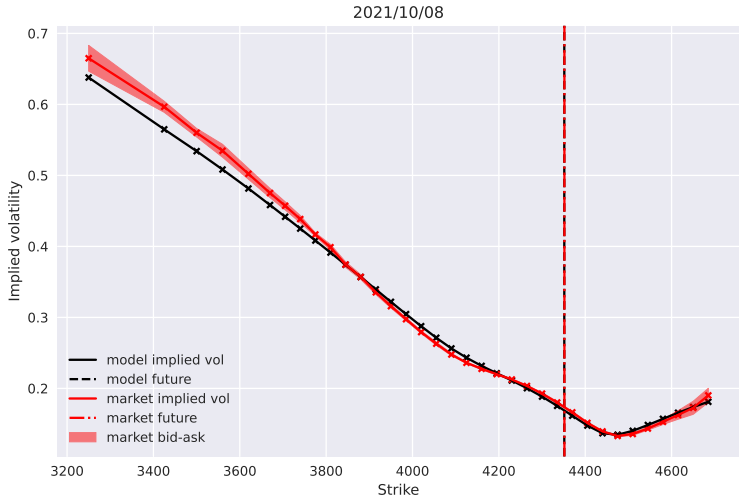
$\theta \leftarrow \textsc{GradientDescent}(\theta, \partial_\theta L, lr)$

return θ

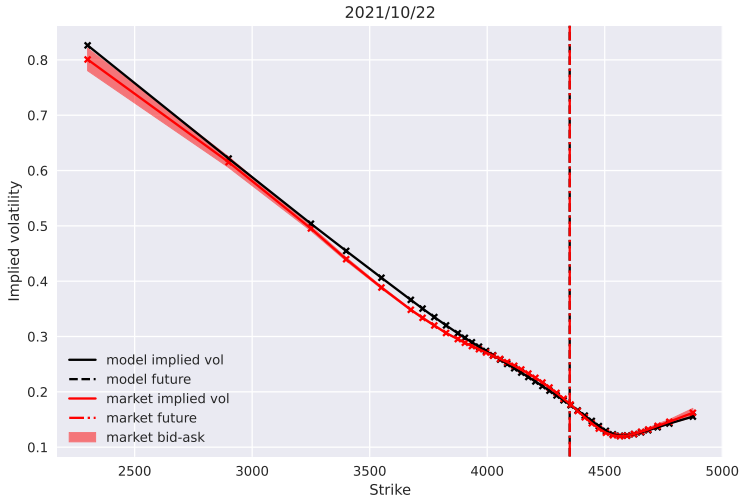
end function

Numerical implementation and data

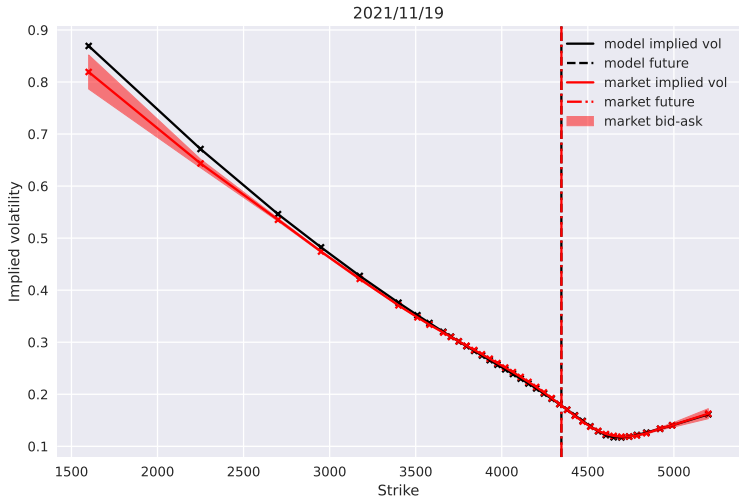
- Number of Monte Carlo paths: $N = 150,000$
- Time step: $\Delta t = 0.5/365$ (half a day)
- Φ_θ is a feedforward NN with $r = 1$ hidden layers of width $l = 16$
- We use the `Adam` algorithm to perform the gradient descent, with learning rate $lr = 0.001$.
- The loss weights are taken as $(w_{f\mathrm{VIX}}, w_{\mathrm{SPX}}, w_{\mathrm{VIX}}) = (30, 2, 3)$.
- The degree of the polynomials for the VIX^2 regression is $d = 8$.
- The nested MC VIX^2 estimator uses $M = 15,000$ nested paths and the MC estimators of VIX payoffs are computed with $N' = 20,000$ trajectories.



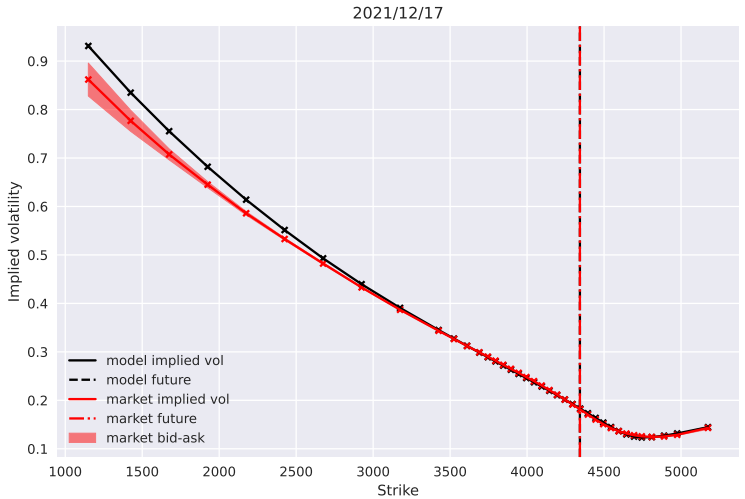
2021/10/08



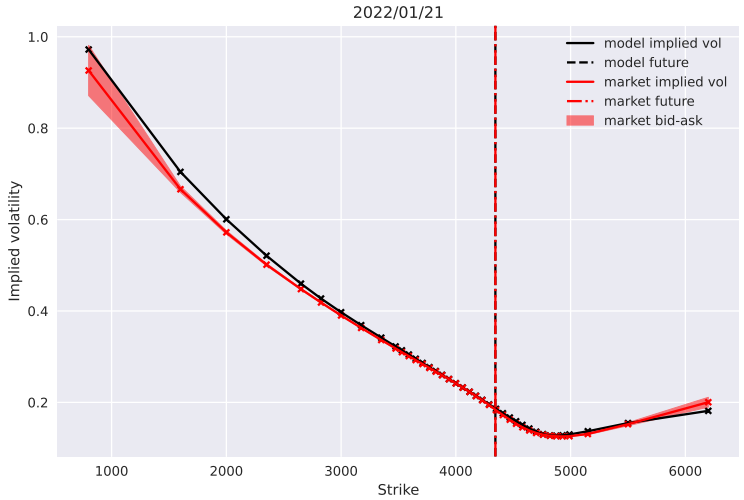
2021/10/22



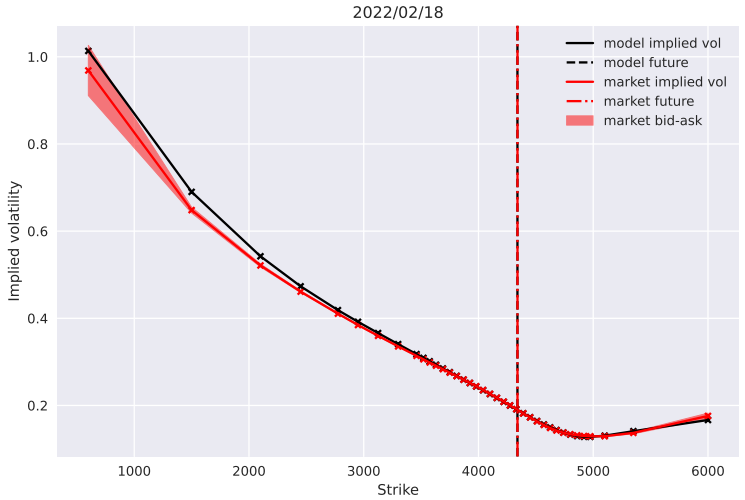
2021/11/19



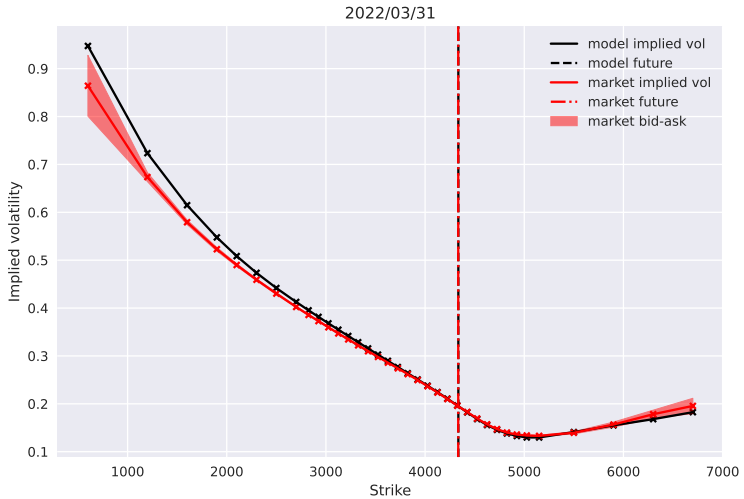
2021/12/17



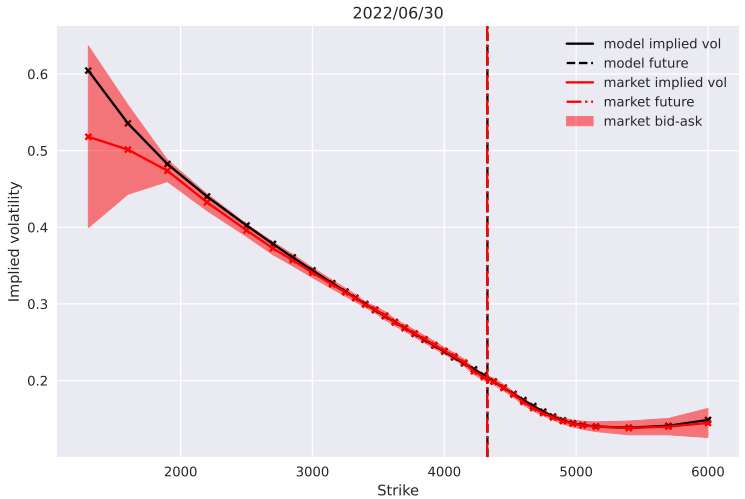
2022/01/21



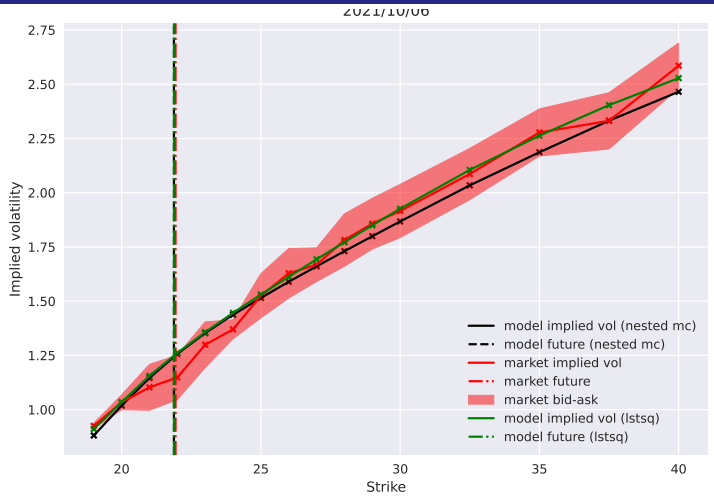
2022/02/18



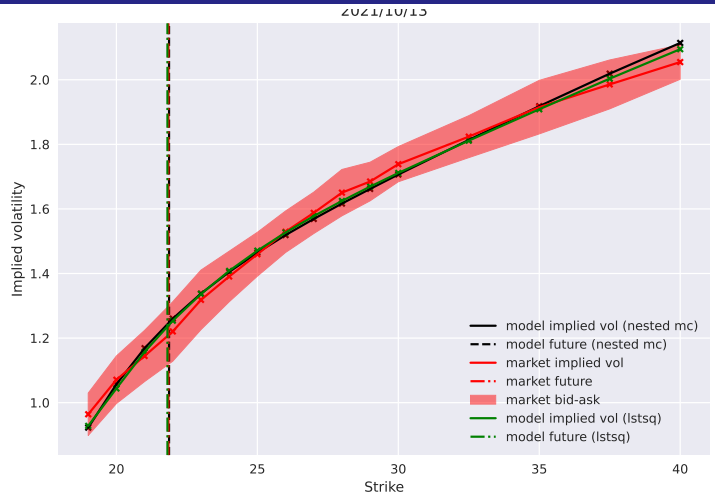
2021/03/31



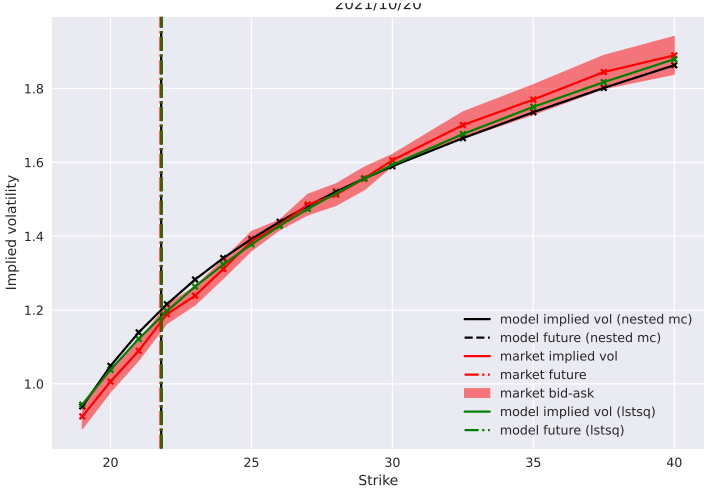
2022/06/30



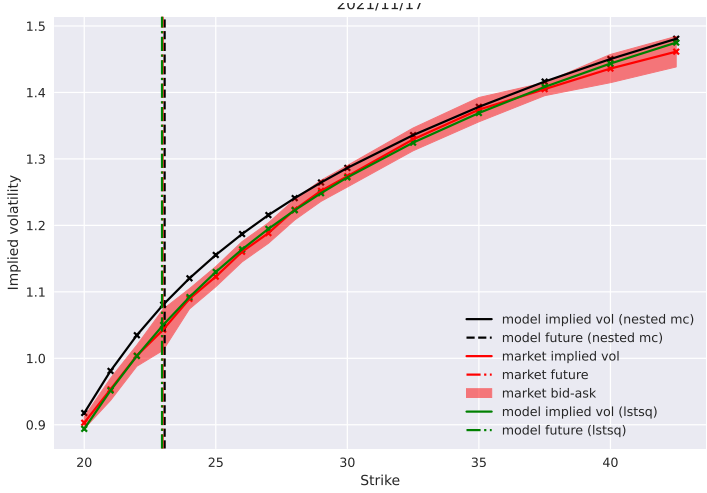
2021/10/06



2021/10/13

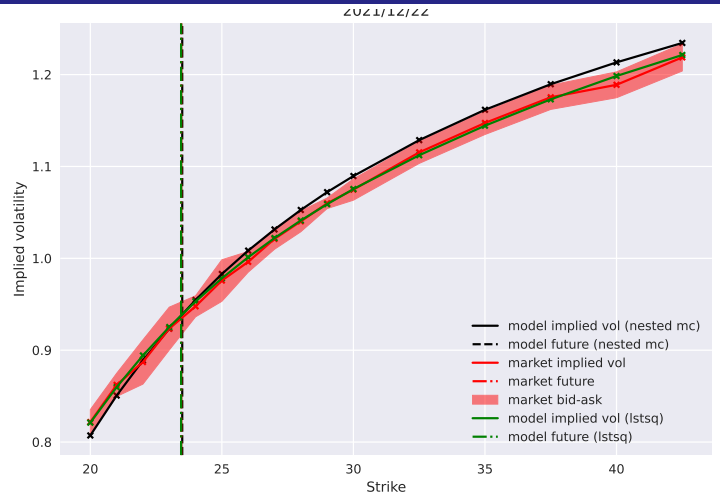


2021/10/20

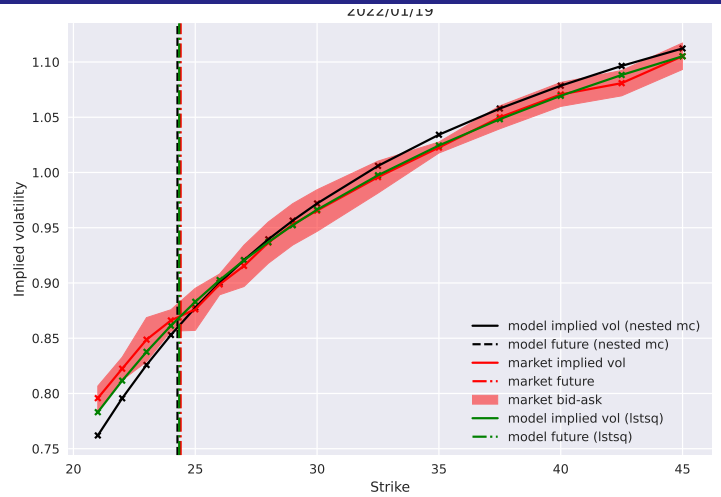


2021/11/17

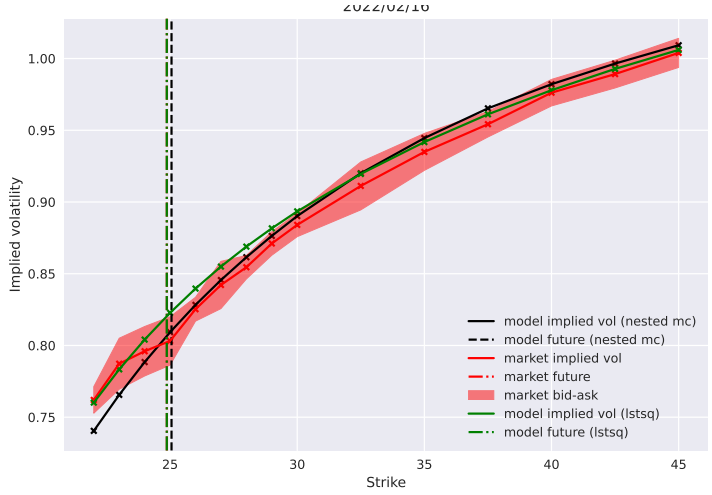
Figure: Calibration of the VIX futures and VIX smiles as of October 1, 2021.



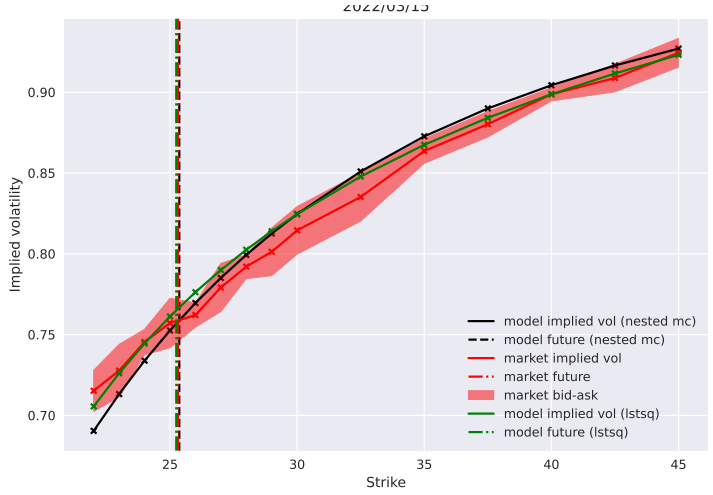
2021/12/22



2022/01/19

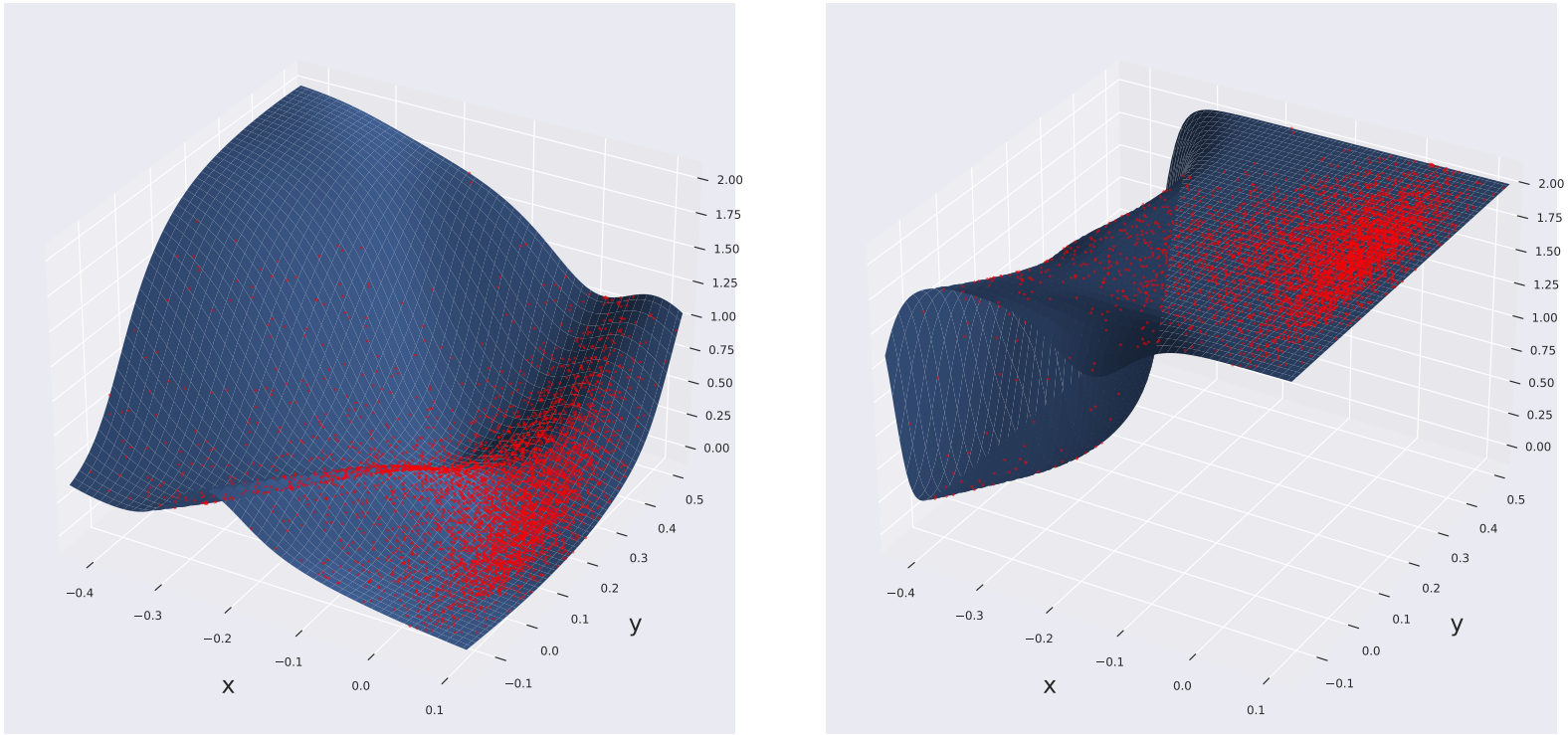


2022/02/16



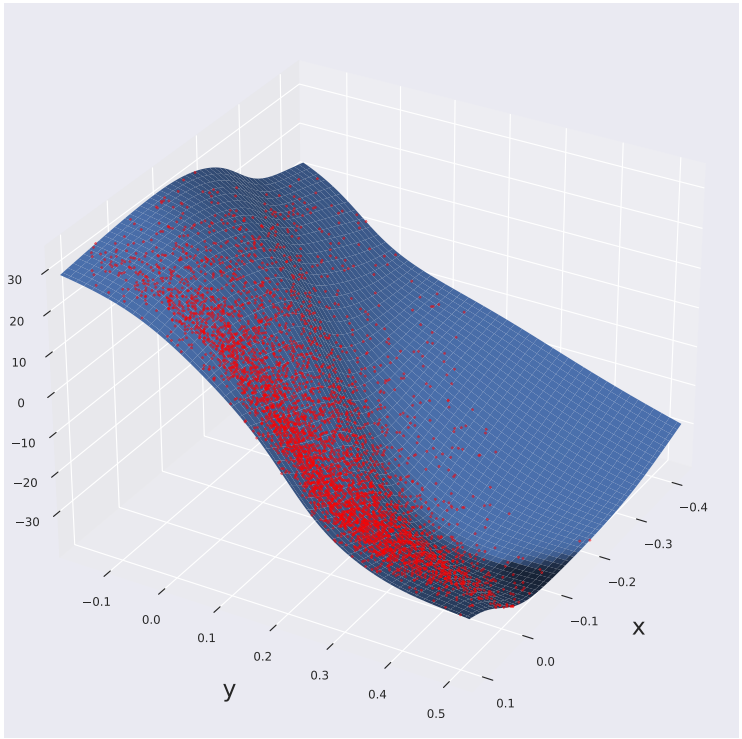
2022/03/15

Figure: Calibration of the VIX futures and VIX smiles as of October 1, 2021.

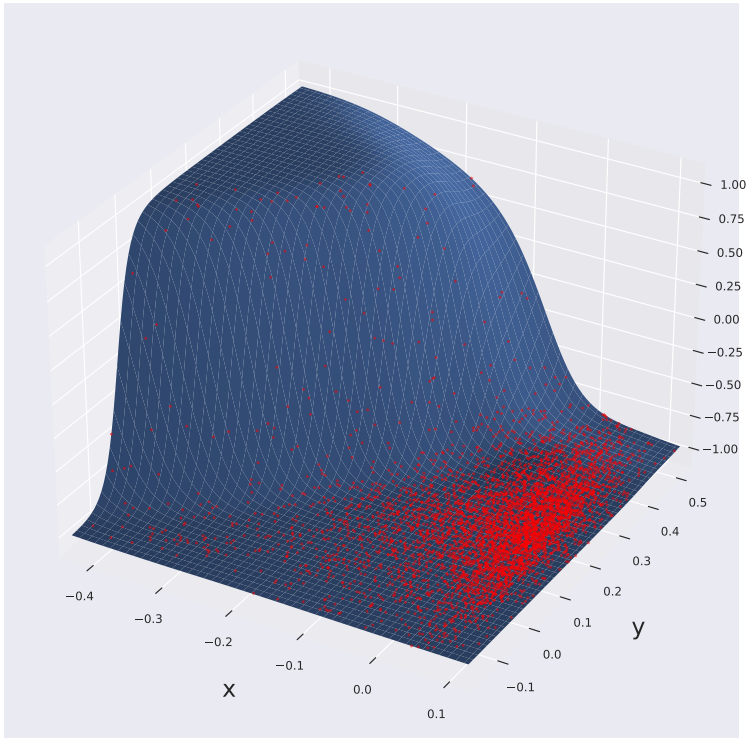


(a) $\sigma_X\,(t,x,y)$ (b) $\sigma_Y\,(t,x,y)$

Figure: Plots of the optimal σ_X, σ_Y at time $t =$ October 13, 2022, as of October 1, 2021.

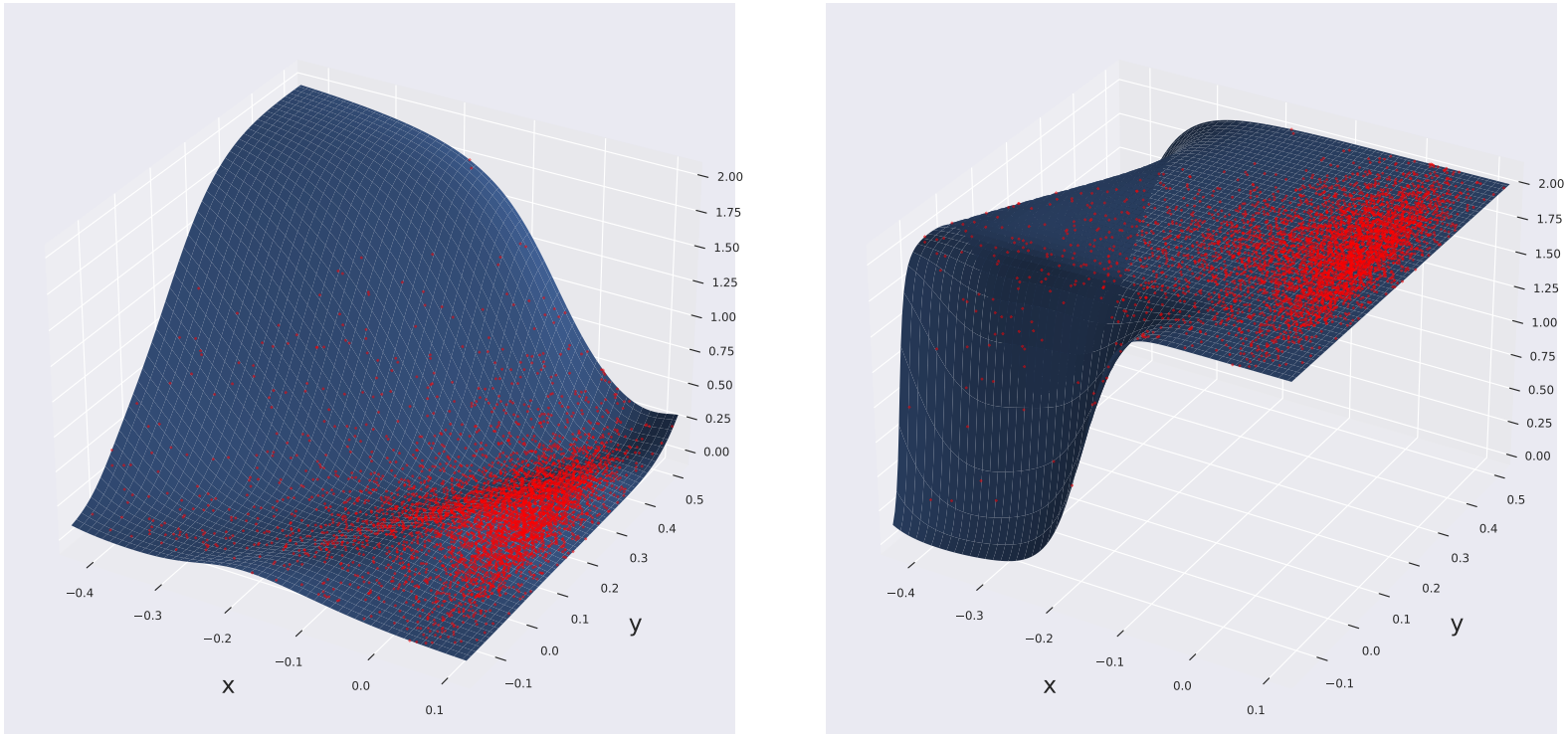


(c) $\mu_Y\,(t,x,y)$



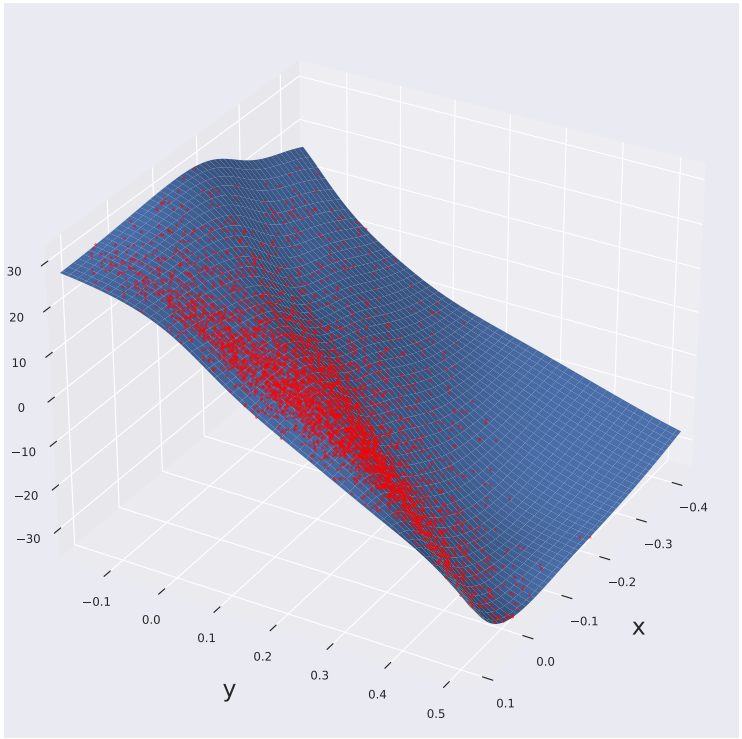
(d) $\rho\,(t,x,y)$

Figure: Plots of the optimal μ_Y, ρ at time $t =$ October 13, 2021, as of October 1, 2021.

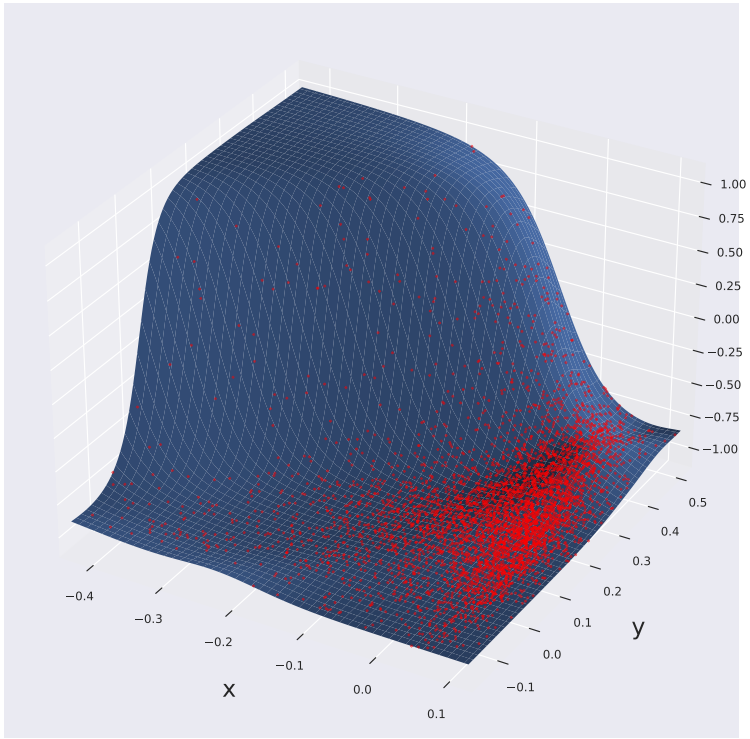


(a) $\sigma_X\,(t,x,y)$ (b) $\sigma_Y\,(t,x,y)$

Figure: Plots of the optimal σ_X, σ_Y at time $t =$ January 13, 2021, as of October 1, 2021.



(c) $\mu_Y\,(t,x,y)$



(d) $\rho\,(t,x,y)$

Figure: Plots of the optimal μ_Y, ρ at time $t =$ January 13, 2022, as of October 1, 2021.

Results

- σ_X is mostly an increasing funct. of Y, so $Y \sim \sigma_X$, but also depends on X
- σ_Y depends on both X and Y, and tends to take small values for small Y, and large values for large Y, so as to produce positive VIX skews.
- We deliberately capped σ_Y to 2 to control the variance of MC estimators.
- The correlation ρ is thus -1 everywhere, so as to match the large negative SPX market skews $\Longrightarrow$ **the model is purely path-dependent**.
- This confirms the findings of Guyon '22, and that PDV models are natural candidates for solving the joint calibration problem.
- The drift μ_Y is mostly negative for positive Y and positive for negative Y: our neural SDE procedure learns that the "volatility" Y **is mean-reverting**.
- Since our NN takes directly (t, x, y) as input and only uses smooth activation functions, the **surfaces are smooth and vary smoothly over time**.
- Calibration time: 36 hours (resp. 3 hour) when we initialize the NN with random weights (resp. the previous business day's weights).

Conclusion

- **A one-factor SLV model can jointly calibrate SPX and VIX smiles and VIX futures for many maturities**, provided enough flexibility is allowed on the SDE coefficients, which we model as neural networks.
- The calibrated model is actually a **one-factor PDV model with a mean-reverting path-dependent factor Y which depends only on past SPX returns**.
- Our work thus illustrates the **expressiveness of neural SDEs** by exactly solving the joint calibration problem for multiple maturities, and provides yet extra reasons to **use PDV models for pricing, hedging, and risk-managing derivatives**.