Deep learning-based methods for implied volatility surfaces

Shuaiqiang Liu

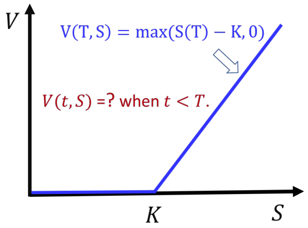
Joint work with Kees Oosterlee, Kees Vuik, etc.

Delft University of Technology & ING Bank, the Netherlands

Outline

A financial product: Option

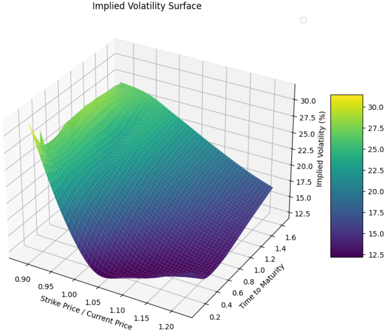
- An option gives a holder the right (not obligation) to trade underlying asset $S(t)$, at a pre-determined price K in future T.
- The option price $V(t, S)$ is the contract fee the holder should pay at time $t < T$ (e.g., starting time $t = 0$).



European call options allow the holder to buy the underlying asset at price K in the maturity time T.

Black-Scholes implied volatility

- The Black-Scholes (BS) model reads $V = BS(\sigma, S_t, K, T-t, r)$ with risk-free interest rate r, volatility σ.
- The BS implied volatility $\sigma^* = BS^{-1}(V^{mkt}, S_t, K, T-t, r)$, given the observed market option price V^{mkt}.
- Implied volatility $\sigma^* := \sigma^*(K, T-t)$ varies over strike prices and time to maturity in practice to form a three-dimenionsal surface.



Implied volatility surface S&P-500 options, November 5th 2023.

Modelling implied volatility surfaces

- Mathematical models: stochastic volatility models (Heston, Bates, etc). Model calibration is required for open parameters.
  - Deep learning volatility [Horvath, et al, 2019].
  - Calibration Neural Networks [Liu, et al, 2019].
- Data-driven methods: deep generative modelling of IVS
  - Variational Autoencoders [Bergeron, et al, 2021]
  - Generative Adversarial Networks [Na, et al, 2023]
  - Diffusion Probabilistic Model [Liu, Ma, et al, 2023]

Mathematical models for volatility surfaces

- Geometric Brownian motion (Black-Scholes model),

$$dS(t) = rS(t)\,dt + \sqrt{\nu}S(t)\,dW_s^{\mathbb{Q}}(t)\,, \quad \nu = \sigma^2,$$

- Considering stochastic volatility (Heston model),

$$\begin{aligned} dS(t) &= rS(t)\,dt + \sqrt{\nu(t)}S(t)\,dW_s^{\mathbb{Q}}(t)\,, \\ d\nu(t) &= \kappa(\bar{\nu} - \nu(t))\,dt + \gamma\sqrt{\nu(t)}\,dW_\nu^{\mathbb{Q}}(t)\,, \\ dW_s^{\mathbb{Q}}(t)\,dW_\nu^{\mathbb{Q}}(t) &= \rho\,dt\,, \end{aligned}$$

- Considering price jumps (Bates model),

$$\begin{aligned} \frac{dS(t)}{S(t)} &= \left(r - \lambda_J \mathbb{E}[e^J - 1]\right)dt + \sqrt{\nu(t)}\,dW_s^{\mathbb{Q}}(t) + \left(e^J - 1\right)dX_{\mathcal{P}}(t)\,, \\ d\nu(t) &= \kappa(\bar{\nu} - \nu(t))\,dt + \gamma\sqrt{\nu(t)}\,dW_\nu^{\mathbb{Q}}(t)\,, \\ dW_s^{\mathbb{Q}}(t)\,dW_\nu^{\mathbb{Q}}(t) &= \rho\,dt\,, \quad X_{\mathcal{P}}(t) \text{ is a Poisson process for jumps.} \end{aligned}$$

Model calibration

- The difference between model value Q and market value Q^* reads,

$$J(\Theta) := \sum_{i=1}^{N} \omega_i ||Q_i - Q_i^*|| + \bar{\lambda}||\Theta||,$$

where Q could be either an option price or implied volatility, N the number of market quotes, $\bar{\lambda}$ a regularization factor.

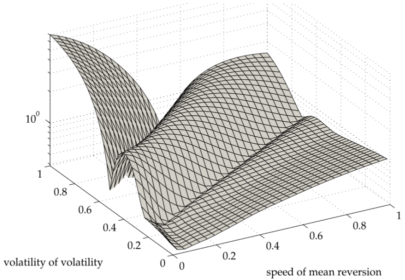
- The objective function,

$$\underset{\Theta \in \mathbb{R}^n}{\operatorname{argmin}} J(\Theta),$$

with n the number of model parameters, e.g., $\Theta := [\rho, \kappa, \gamma, \bar{\nu}, \nu_0]$ in Heston, $\Theta := [\rho, \kappa, \gamma, \bar{\nu}, \nu_0, \lambda_J, \mu_J, \sigma_J]$ in Bates.

Challenges of model calibration

- Model calibration is computationally expensive and slow.
- The objective functions are often non-convex (local minima).



Multiple minima when calibrating Heston (Gilli and Schumann, 2011).

- Calibration Neural Network (CaNN)[1], a deep learning-based framework for fast model calibration, has been developed.

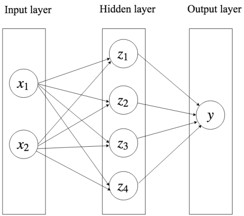
[1] S. Liu, et al.(2019) A neural network-based framework for financial model calibration, J. of Mathematics in Industry

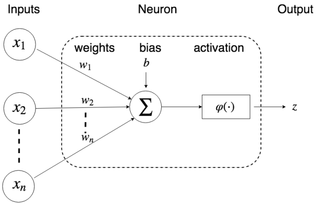
Artificial Neural Networks (ANNs)

- ANNs are a composite function mathematically,

$$F(x|\theta) = f^{(H)}(...f^{(2)}(f^{(1)}(x;\theta^{(1)});\theta^{(2)});...\theta^{(H)})$$

  where $\theta = (\mathbf{W}_i, \mathbf{b}_i)$, $\mathbf{W}_i$ weight matrix and $\mathbf{b}_i$ bias vector.
- A hidden neuron follows $z_j^{(h)} = \varphi^{(h)}\left(\sum_i w_{ij}^{(h)} z_i^{(h-1)} + b_j^{(h)}\right)$.





Training ANNs

- Stochastic Gradient Descent (SGD) algorithm to update the weights and biases during the training phase,

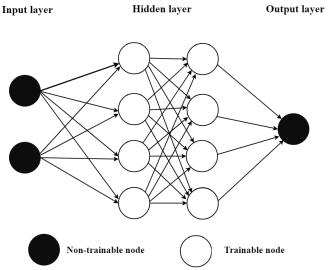
$$\begin{cases} \mathbf{W}_{i+1} \leftarrow \mathbf{W}_i - \eta(i)\dfrac{\partial L}{\partial \mathbf{W}}, \\ \mathbf{b}_{i+1} \leftarrow \mathbf{b}_i - \eta(i)\dfrac{\partial L}{\partial \mathbf{b}}, \\ \eta \text{ learning rate}, L \text{ loss function}, i = 0, 1, 2, ... \end{cases}$$

The variants include Adam, RMSprop, etc.

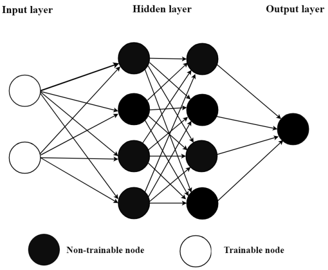
- The goal of training the ANN is optimizing hidden parameters θ to minimize the loss function.

Calibration neural networks

- CaNN consists of three phases, training/prediction/calibration.
- The training/prediction phases learn behaviours of numerical solvers, while the calibration phase inverts the trained ANN.
- The three phases are viewed as a whole, and the difference is to simply open/close the learnable units in input/hidden/ouput layers.



(c) Training phase (offline)



(d) Calibration phase (online)

Optimization algorithms for CaNN

- The training phase: gradient-based stochastic local optimizer, e.g. Adam.
- The calibration phase: gradient-free global optimizer, e.g. Differential evolution (DE).

Differential Evolution method

1. Initialization: Randomly generate a population with N_p individuals,

$$(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{N_p})$$

2. Mutation: Add a randomly sampled difference to each individual,

$$\boldsymbol{\theta}'_i = \boldsymbol{\theta}_a + F \cdot (\boldsymbol{\theta}_b - \boldsymbol{\theta}_c)$$

where, i represents the i-th candidates.

3. Crossover: Filter out some samples by a user-defined crossover possibility $Cr \in [0, 1]$,

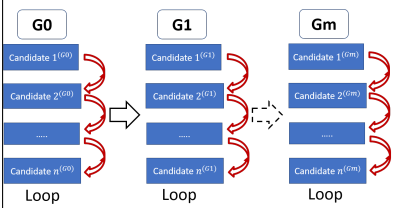
$$\boldsymbol{\theta}''_i = \begin{cases} \boldsymbol{\theta}'_i, \text{ if } p_i \leq Cr \\ \boldsymbol{\theta}_i, \text{ otherwise} \end{cases}$$

4. **Selection**: Compare each new trial candidate with the corresponding target individual on the objective function,

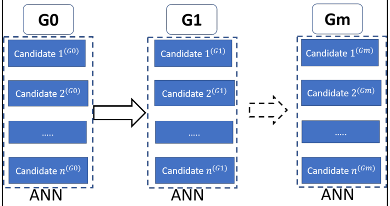
$$\boldsymbol{\theta}_i \leftarrow \begin{cases} \boldsymbol{\theta}''_i, \text{ if } g(\boldsymbol{\theta}''_i) \leq g(\boldsymbol{\theta}_i) \\ \boldsymbol{\theta}_i, \text{ otherwise.} \end{cases}$$

Accelerated CaNN

- A generation of candidates enter into the ANN simultaneously.



(e) Conventional DE



(f) Parallel DE

Calibrating Heston model

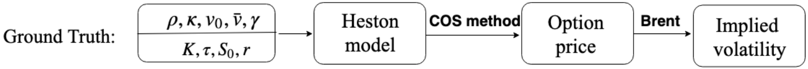
- The Heston option pricing PDE reads,

$$\begin{aligned}\frac{\partial V}{\partial t} \quad &+ \quad rS\frac{\partial V}{\partial S} + \kappa(\bar{\nu} - \nu_t)\frac{\partial V}{\partial \nu_t} + \frac{1}{2}\nu_t S^2 \frac{\partial^2 V}{\partial S^2} \\ &+ \quad \rho\gamma S\nu_t \frac{\partial^2 V}{\partial S \partial \nu_t} + \frac{1}{2}\gamma^2 \nu_t \frac{\partial^2 V}{\partial \nu_t^2} - rV = 0.\end{aligned}$$

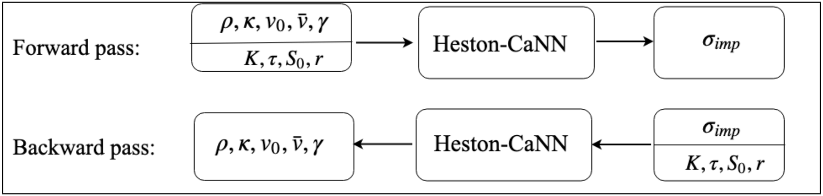
  where $V = V(t, S, \nu_t; K, T)$ is the option price at time t, with suitable terminal conditions.
- Calibrating Heston model is to estimate five parameters, correlation coefficient ρ, long term variance $\bar{\nu}$, reversion speed κ, volatility of volatility γ, initial variance ν_0, given an implied volatility surface.

Heston-CaNN

- Heston-CaNN consists of two stages, a forward pass and a backward pass,



- The forward pass produces a fast ANN solver to solve Heston. The backward pass (the calibration phase) yields input model parameters to match the market data.



Performance of Heston-CaNN

- Calibration to 35 market quotes (7 strikes and 5 maturity time).
- Heston-CaNN performance over 15,625 test cases.

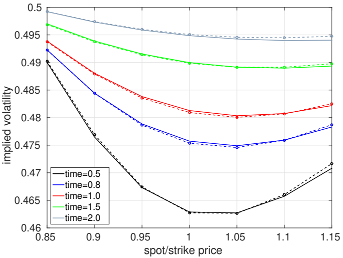
Deviation from true $\boldsymbol{\Theta}^*$		Averaged Cost/Error	
$\lvert\nu_0^\dagger - \nu_0^*\rvert$	4.39×10^{-4}	CPU time (seconds)	0.85
$\lvert\bar{\nu}^\dagger - \bar{\nu}^*\rvert$	4.54×10^{-3}	GPU time (seconds)	0.48
$\lvert\gamma^\dagger - \gamma^*\rvert$	3.28×10^{-2}	Function evaluations	$193,249$
$\lvert\rho^\dagger - \rho^*\rvert$	4.84×10^{-2}	Data points	35
$\lvert\kappa^\dagger - \kappa^*\rvert$	4.88×10^{-2}	Calibration error $J(\Theta)$	2.52×10^{-6}

A higher dimensional case: Bates-CaNN

- Calibrate eight parameters using Bates-CaNN to deal with more complex implied volatility surfaces.

- The dotted implied volatility curves are from Bates-CaNN, and the solid curves are "observed" in the market.

Parameters	Search space	True	Calibrated
Intensity of jumps, λ_J	[0, 3.0]	1.0	1.06
Mean of jumps, μ_J	[0, 0.4]	0.1	0.09
Variance of jumps, ν_J^2	[0, 0.3]	0.4^2	0.15
Correlation, ρ	[-0.9, 0.0]	-0.3	-0.22
Reversion speed, κ	[0.1, 3.0]	1.0	0.60
Long average variance, $\bar{\nu}$	[0.01, 0.5]	0.1	0.13
Volatility of volatility, γ	[0.01, 0.8]	0.7	0.78
Initial variance, ν_0	[0.01, 0.5]	0.1	0.10
Total Squared Error	-	-	4.9×10^{-6}
Function evaluation	-	-	842,800
Time(seconds)	-	-	1.8



CaNN for rough Heston model

There are six parameters to calibrate in rough Heston model [2].

$$dS_t = S_t\sqrt{v_t}dW_t,$$

$$v_t = v_0 + \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1}\gamma(\theta - v_s)ds + \frac{1}{\Gamma(\alpha)}\int_0^t (t-s)^{\alpha-1}\gamma\nu\sqrt{v_s}dB_s, \quad (2)$$

with $\alpha \in (1/2, 1)$ determining the roughness of the volatility process, where $\alpha = H + 1/2$. H is the Hurst parameter.

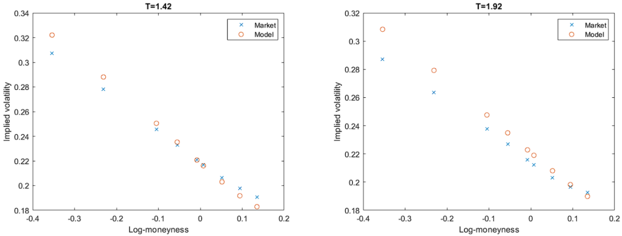


Figure (19) Market vs. Model implied volatility smiles using the rough Heston-ANN calibration

[2] Erkan K. E.(2020). European option pricing under the rough Heston model using the COS method, MSc thesis, TU Delft.

CaNN on real market data

- The performance of CaNN on real market data[3].

Review of Derivatives Research (2022) 25:109–136
https://doi.org/10.1007/s11147-021-09183-7

Check for updates

Deep calibration of financial models: turning theory into practice

Patrick Büchel[1] · Michael Kratochwil[2] · Maximilian Nagl[3] · Daniel Rösch[3]

Accepted: 15 July 2021 / Published online: 17 August 2021

✉ Maximilian Nagl
maximilian.nagl@ur.de

Patrick Büchel
patrick.buechel@commerzbank.com

Michael Kratochwil
michael.kratochwil@nagler-company.com

Daniel Rösch
daniel.roesch@ur.de

1 Commerzbank AG, Mainzer Landstraße 157, 60327 Frankfurt am Main, Germany

2 Dr. Nagler & Company GmbH, Maximilianstraße 47, 80538 Munich, Germany

3 Universität Regensburg, Chair of Statistics and Risk Management, Universitätsstraße 31, 93040 Regensburg, Germany

COVID-19 pandemic in spring 2020. The daily swaption data is available for different expiry tenor, swap tenor and strike values:

- Option Tenor: 1M, 3M, 6M, 9M, 1Y, 2Y, 5Y, 10Y, 15Y, 20Y
- Swap Tenor: 1Y, 2Y, 5Y, 10Y, 15Y, 20Y, 30Y
- Strike (ATM ± bp): 0, 12.5, 25, 50, 100, 150, 200

On each trading day, we observe valid prices for about 800 swaptions. This amounts to a total number of more than 350,000 price observations. In practical applications,

the CaNN framework does provide **comparable** calibration results even in extreme and unusual market situations in a **faster** and computationally **more efficient** manner.

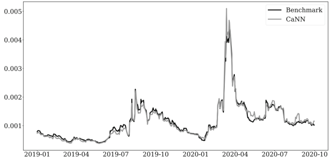


Fig. 8 Sum of squared errors over trading days. Note: This figure shows the sum of squared errors of trading days for the whole time span. The grey line corresponds to the SSE using the CaNN approach, whereas the black line coincides with the SSE of the benchmark implementation.

the period from June 2019 to August 2019 or the early months of 2019. The largest deviation between the CaNN and the benchmark implementation can be observed during the COVID-19 period in the March 2020. Nevertheless, the daily performance of both approaches does not differ significantly even in this stressed market environment. Hence, the CaNN framework does provide comparable calibration results even in extreme and unusual market situations in a faster and computationally more efficient manner. Furthermore, the very good results for the out-of-time period (May to September 2020) indicate that the performance of the CaNN framework does not depend on including current market data during training.

[3] Buchel, et al. (2021) Deep calibration of financial models: turning theory into practice. Review of Derivatives Research.

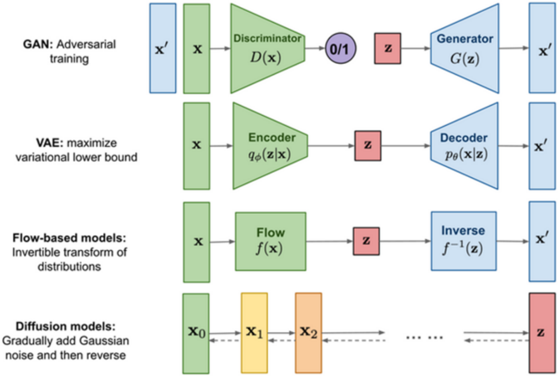
Diffusion Probabilistic Model for Implied Volatility Surface Generation and Completion

- Mathematical models: stochastic volatility models (Heston, Bates, etc). Model calibration is required for open parameters.
  - Deep learning volatility [Horvath, et al, 2019].
  - Calibration Neural Networks [Liu, et al, 2019].
- Data-driven methods: deep generative modelling of IVS,
  - Variational Autoencoders [Bergeron, et al, 2021].
  - Generative Adversarial Networks [Na, et al, 2023].
  - Diffusion Probabilistic Model [Liu, Ma, et al, 2023][4].

[4] Ma, X. (2023). Diffusion Probabilistic Model for Implied Volatility Surface Generation and Completion, MSc Thesis, TU Delft

Overview of generative deep learning models

- Generative Adversarial Networks (GAN).
- Variational Autoencoder (VAE).
- Flow-based models.
- Diffusion probabilistic models.

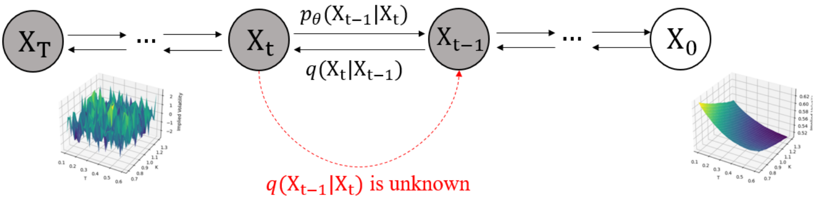


Denoising Diffusion Probabilistic Models

- Diffusion models employ neural networks to remove noise,

  Forward process: $\mathrm{d}\mathbf{x} = f(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W\,, \mathbf{x}(0) = \mathbf{x}_0$

  Reverse process: $\mathrm{d}\mathbf{x} = \left[f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log q_t(\mathbf{x})\right]\mathrm{d}t + g(t)\,\mathrm{d}W\,, \mathbf{x}(T) = \mathbf{x}_T$



- Denoising Diffusion Probabilistic Models (DDPM) [Ho, et al, 2020] use the Markov chain of forward (reverse) diffusion process,

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}W\,, \mathbf{x}(0) = \mathbf{x}_0,$$

  where $\beta(t) := \beta_t \in (0, 1)$ is a user-defined hyperparameter.

Denoising Diffusion Probabilistic Models

Training	Generating
1: Input: implied volatility surfaces $q(\mathbf{x}_0)$.	1: Input: already trained networks $\boldsymbol{\epsilon}_\theta$.
2: **repeat**	2: $\mathbf{x}_M \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: Select $\mathbf{x}_0$ from $q(\mathbf{x}_0)$	3: **for** $t = M, \ldots, 1$ **do**
4: $t \sim \text{Uniform}(\{1, \ldots, M\})$	4: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
5: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	5: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
6: Stochastic Gradient Descent $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$	6: **end for**
7: **until** converged	7: **return** $\mathbf{x}_0$

- $\mathbf{x}_t := \mathbf{x}(t)$ represents the intermediate result at time t.
- $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ represents a neural network with hidden parameters θ.
- The hyper-parameters $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{M} \alpha_i$.

Comparison between generating and completing IVS

Given already trained neural networks $\boldsymbol{\epsilon}_\theta$,

- IVS Generation: starting from a random IVS $\mathbf{x}_{t=M}$, go backward through reparameterization

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}. \tag{3}$$

- IVS Completion: starting from a random IVS $\mathbf{x}_{t=M}$ and an incomplete IVS $\hat{\mathbf{x}}_0$,

$$\boldsymbol{x}_{t-1} = \boldsymbol{m} \odot \boldsymbol{x}_{t-1}^{known} + (\mathbf{1} - \boldsymbol{m}) \odot \boldsymbol{x}_{t-1}^{unknown}, \tag{4}$$

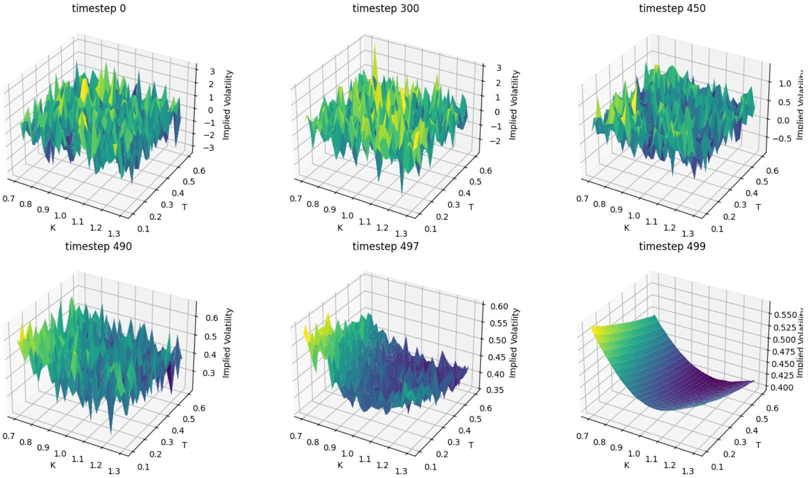
where matrix $\boldsymbol{m}$ locates missing data points,
$\mathbf{x}_{t-1}^{known} = \sqrt{\alpha_t}\hat{\mathbf{x}}_0 + (1-\alpha_t)\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and
$\mathbf{x}_{t-1}^{unknown} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Completing partial IVS with DDPM

Algorithm Completing partial IVS with DDPM

1: Input: already trained NN $\boldsymbol{\epsilon}_\theta$ and partial IVS $\hat{\mathbf{x}}_0$.
2: Sample initial state $\mathbf{x}_M \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
3: **for** $t = M, \dots, 1$ **do**
4: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad \mathbf{x}_{t-1}^{unknown} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
6: $\quad \mathbf{x}_{t-1}^{known} = \sqrt{\alpha_t}\hat{\mathbf{x}}_0 + (1 - \alpha_t)\boldsymbol{\epsilon}$
7: $\quad \mathbf{x}_{t-1} = \mathbf{m} \odot \mathbf{x}_{t-1}^{known} + (\mathbf{1} - \mathbf{m}) \odot \mathbf{x}_{t-1}^{unknown}$
8: **end for**
9: Output: a completed IVS $\mathbf{x}_0$.

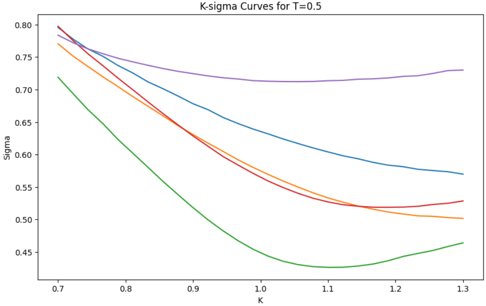
The process of generating IVS with DDPM



Intermediate time steps to generate an implied volatility surface.

Generating IVS

- Implied volatility curves (smile, etc) generated by DDPM

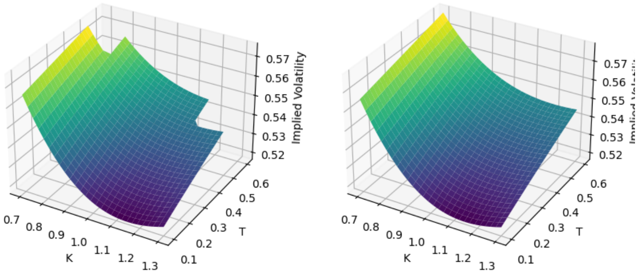


- Statistics distance between generated and historical IVS:

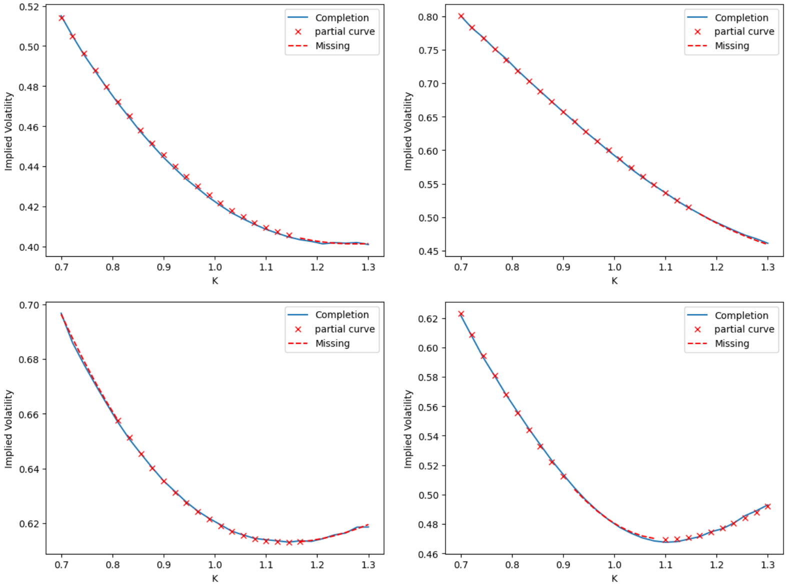
Timestep	1-Wasserstein
0	611.23
300	461.37
450	75.52
499	3.68

Completing partial IVS

- Comparing partial IVS (Left) with completed IVS (Right) by DDPM:



Completing partial IVS (Cont')



Conclusions and ongoing work

Conclusions:

- CaNN provides a fast model calibration framework for stochastic volatility models.
- DDPM can produce high-quality synthetic and complete partial implied volatility surfaces.
- The generative AI approach, diffusion models, can complement the mathematical modelling approach in processing implied volatility surfaces.

Ongoing work:

- Explicitly incorporate the arbitrage-free conditions into the DDPM generation process for implied volatility surfaces, aiming to enhance financial consistency and reliability.