Deep Smoothing of Implied Volatility Surface Technical Report

Sina Baghal
siinabaghal@gmail.com

August 2024

Abstract

This project implements the methodologies presented in the paper [ATV20]. The main idea is to use feedforward neural networks as a corrective tool to modify the prior models considered for volatility surfaces. By letting

$$\omega(k,\tau;\theta) := \omega_{\text{prior}}(k,\tau;\theta_{prior}) \cdot \omega_{\text{nn}}(k,\tau;\theta_{nn}), \qquad \text{(Implied Variance)}$$

we enrich the space of parameters used for fitting the volatility surface. Here θ_{prior} and θ_{nn} are two disjoint set of parameters. Furthermore, to ensure our volatility surface is free of arbitrage, we use the ideas in [Rop10] which argues that if the following are satisfied then the call price surface is free of Calendar & Butterfly arbitrage resp.

$$\ell_{cal} = \partial_\tau \omega(k,\tau) \geq 0$$

$$\ell_{but} = \left(1 - \frac{k\partial_k\omega(k,\tau)}{2\omega(k,\tau)}\right)^2 - \frac{\partial_k\omega(k,\tau)}{4}\left(\frac{1}{\omega(k,\tau)} + 0.25\right) + \frac{\partial^2_{kk}\omega(k,\tau)}{2} \geq 0$$

Another condition required to guarantee a free-arbitrage VS is the large moneyness behaviour which states that $\sigma^2(k,\tau)$ is linear for $k \to \pm\infty$ for every $\tau > 0$. [Rop10] achieves this by imposing $\frac{\sigma^2(k,\tau)}{|k|} < 2$ which in turn is achieved by minimizing the following

$$\frac{\partial^2\omega(k,\tau)}{\partial k \partial k} \qquad \text{(Asymptotic Behaviour)}$$

We use the above three constraints to shape the loss function utilized in training the implied variance ω. As for learning rate scheduling, we use a slightly different approach than [ATV20]; we employ model weights perturbation along with a divergence handling scheme.

Numerical results show that our enhanced model, incorporating a neural network with the loss function (Implied Variance), fits the Bated model data perfectly and produces an arbitrage-free volatility surface. Python code for this project can be found at the following link:

https://github.com/sinabaghal/sinabaghal.github.io/tree/master/files/codes/deepsmoothing

Contents

1 Notation and Initial Values

European call options with the following table of notation and values are used:

Parameter	**Value/Definition**
Spot Price	$spot = \$1$
Strike	K
Interest Rate	$rate = 0.0$
Dividend Rate	$q = 0.0$
Forward Price	F_t
Forward Log Moneyness	$k = \log \frac{K}{F_t}$
Implied Volatility	$\sigma(k, \tau)$
Implied Variance	$\omega(k, \tau) := \sigma(k, \tau)^2 \tau$

Table 1: Parameters and Definitions

2 Bates Model

Bates dynamic is as follows:

$$\frac{dS_t}{S_t} = (r-\delta)\,dt + \sqrt{V_t}\,dW_{1t} + dN_t$$
$$dV_t = \kappa(\theta - V_t)\,dt + \sigma\sqrt{V_t}\,dW_{2t}$$

$dW_{1t}dW_{2t} = \rho dt$ and N_t is a compound Poisson process with intensity λ and independent jumps J with

$$\ln(1+J) \sim \mathcal{N}\left(\ln(1+\beta) - \frac{1}{2}\alpha^2, \alpha^2\right)$$

2.1 Characteristic Function

The characteristic function of the log-strike in Bates model is given by:

$$\begin{aligned}\phi(\tau,u) = &\exp\left(\tau\lambda\cdot\left(e^{-0.5\alpha^2u^2+iu\left(\ln(1+\beta)-0.5\alpha^2\right)}-1\right)\right)\\ &\cdot\exp\left(\frac{\kappa\theta\tau(\kappa-i\rho\sigma u)}{\sigma^2}+iu\tau(rate-\lambda\cdot\beta)+iu\cdot\log spot\right)\cdot\left(\cosh\frac{\gamma\tau}{2}+\frac{\kappa-i\rho\sigma u}{\gamma}\cdot\sinh\frac{\gamma\tau}{2}\right)^{-\frac{2\kappa\theta}{\sigma^2}}\\ &\cdot\exp\left(-\frac{(u^2+iu)v_0}{\gamma\coth\frac{\gamma\tau}{2}+\kappa-i\rho\sigma u}\right).\end{aligned}$$

2.2 Fast Fourier Transform

Option price calculation for Bates model is done using FFT. Following [CM99], we apply a smoothing technique to be able to compute the FFT integral: Recall that the option value is given by

$$C_\tau(k) = spot\cdot\Pi_1 - Ke^{-rate\cdot\tau}\cdot\Pi_2$$

where

$$\Pi_1 = \frac{1}{2}+\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left[\frac{e^{-iu\ln(K)}\phi_\tau(u-i)}{iu\phi_\tau(-i)}\right]du$$
$$\Pi_2 = \frac{1}{2}+\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left[\frac{e^{-iu\ln(K)}\phi_\tau(u)}{iu}\right]du$$

Since the integrand is singular at the required evaluation point $u = 0$, FFT cannot be used to evaluate call price $C_\tau(k)$. To offset this issue, we consider modified call price $c_\tau(k) := \exp(\alpha k)C_\tau(k)$ for $\alpha > 0$. Denote Fourier transform of $c_\tau(k)$ by

$$\Psi_\tau(v) = \int_{-\infty}^\infty e^{ivk}c_\tau(k)\,dk \Rightarrow C_\tau(k) = \frac{\exp(-\alpha k)}{\pi}\int_0^\infty e^{-ivk}\Psi_\tau(v)\,dv$$

It can be shown that

$$\Psi_\tau(v) = \frac{e^{-r\tau}\phi_\tau(v-(\alpha+1)i)}{\alpha^2+\alpha-v^2+i(2\alpha+1)v}$$

2.2.1 FFT Setup

We setup FFT calculation as below

- Log strike levels ranges from $-b$ to b where

$$b = \frac{Ndk}{2}$$

- $\Psi_\tau(u)$ are computed at the following v values

$$v_j = (j-1)du \text{ for } j = 1, \cdots, N$$

- Option prices are computed at the following k values

$$k_u = -b + dk(u-1) \text{ for } u = 1, \cdots, N$$

- To apply FFT, we need to set

$$dk \cdot du = \frac{2\pi}{N}$$

- Simposon weights are used

$$3 + (-1)^j - \delta_{j-1} \text{ for } j = 1, \cdots, N$$

Having this setup ready, call prices are obtained as follows:

$$C(k_u) = \frac{\exp(-\alpha k_u)}{\pi} \sum_{j=1}^{N} e^{-i\frac{2\pi}{N}(j-1)(u-1)} e^{ibv_j} \Psi(v_j) \frac{\eta}{3} \left(3 + (-1)^j - \delta_{j-1}\right)$$

3 Loss Function

We define four different loss functions and construct the total loss function as a linear combination of these four with coefficients being the penalty parameters. The first loss function is the prediction error which is defined as below

$$\mathcal{L}_0(\theta) \qquad \text{(Prediction Error)}$$
$$= \sqrt{\frac{1}{|\mathcal{I}_0|} \sum_{(\sigma,k,\tau)\in\mathcal{I}_0} (\sigma - \sigma_\theta(k,\tau))^2} + \frac{1}{|\mathcal{I}_0|} \sum_{(\sigma,k,\tau)\in\mathcal{I}_0} \frac{|\sigma - \sigma_\theta(k,\tau)|}{\sigma}$$

Here $\mathcal{I}_0$ is the set of log-moneyness, implied volatility and maturities for each observed market option. For future use, denote

$$\mathcal{T}_0 = \{\tau : (0,\tau) \in \mathcal{I}_0\} \qquad (\mathcal{T}_0)$$

The other three loss functions are auxiliary and consequently we need to introduce auxiliary maturities and log moneyness values. This is to ensure desired features for the constructed implied variance ω across unobserved market data. Define

$$\mathcal{T}_{aux} = \left\{ \exp(x) : x \in \left[\log\left(\frac{1}{365}\right), \max(\log(\tau_{\max}(\mathcal{I}_0) + 1)) \right]_{100} \right\}$$
$$k_{aux} = \left\{ x^3 : x \in \left[-(-2k_{\min})^{1/3}, (2k_{\max})^{1/3} \right]_{100} \right\}$$
$$\mathcal{I}_{aux} = \{(k, \tau) : k \in k_{aux}, \tau \in \mathcal{T}_{aux}\}$$

where *e.g.,* $k_{\max} = k_{\max}(\mathcal{I}_0)$. Calendar and Butterfly loss functions are then defined as

$$\mathcal{L}_{cal}(\theta) = \frac{1}{|\mathcal{I}_{Aux}|} \sum_{(k,t)\in\mathcal{I}_{Aux}} \max\left(0, -\ell_{cal}(k, \tau)\right) \qquad \text{(Calendar Loss)}$$

$$\mathcal{L}_{but}(\theta) = \frac{1}{|\mathcal{I}_{Aux}|} \sum_{(k,t)\in\mathcal{I}_{Aux}} \max\left(0, -\ell_{but}(k, \tau)\right) \qquad \text{(Butterfly Loss)}$$

For large moneyness behaviour, we set

$$\mathcal{L}_{asym}(\theta) = \frac{1}{|\mathcal{I}_{asym}|} \sum_{(k,\tau)\in\mathcal{I}_{asym}} \left| \frac{\partial^2 \omega_(k, \tau)}{\partial k \partial k} \right|$$

where

$$\mathcal{I}_{asym} = \{(k, \tau) : k \in \{6k_{\min}, 4k_{\min}, 4k_{\max}, 6k_{\max}\}, \tau \in \mathcal{T}_{Aux}\}$$

3.1 Around the Money

We prefer that the prior component in (Implied Variance) gives the best possible fit for ATM and neural network fixes prior's limitations for OTM. To this end, the following loss function is considered.

$$\mathcal{L}_{atm}(\theta) = \frac{1}{|\mathcal{I}_{\text{atm}}|} \left(\sum_{(k,\tau)\in\mathcal{I}_{\text{atm}}} (1 - \omega_{nn}(k, \tau; \theta_{nn}))^2 \right)^{1/2}$$

Here

$$\mathcal{I}_{\text{atm}} = \{(0, \tau) : \tau \in \mathcal{T}_{Aux}\}.$$

3.2 Total Loss Function

Total loss function is constructed as below.

$$\mathcal{L}_{Tot}(\theta) = \mathcal{L}_0(\theta) + \lambda_{but}\mathcal{L}_{but} + \lambda_{cal}\mathcal{L}_{cal} + \lambda_{atm}\mathcal{L}_{atm} \qquad \text{(Total Loss)}$$

Regularization parameters $\lambda_{but}, \lambda_{cal}, \lambda_{atm}$ are tunable. In our experiments, we consider

Parameter	Value
λ_{atm}	0.1
λ_{but}	4
λ_{cal}	4

Table 2: Loss Function Regularization Parameters

4 ATM Total Variance

Prior model is expected to be a first-order approximation of the volatility surface and therefore it is important that the prior reproduces the ATM values accurately. Note ATM term structure is observable through market data for $\tau \in \mathcal{T}_0$ ($\mathcal{T}_0$), we construct ATM variance for every $\tau \in \mathcal{T}_{aux}$ as explained below.

Step A: Ensure $\omega_{atm}(\tau_2) \geq \omega_{atm}(\tau_1)$ for every $\tau_2 \geq \tau_1$. To this end, we solve the following optimization problem using CVX.

$$\sum_{i=0}^{|\mathcal{T}_0|} \left(z_i - \omega_{atm}(\tau_i)\right)^2 \quad \text{s.t.} \quad z_{|\mathcal{T}_0|} \geq \cdots \geq z_0$$

Step B: We interpolate results from Step A using scipy with degree of the smoothing spline equal to 3.

Step C: We next fit a degree five polynomial to the result from Step B. This way, we have access to both $\omega_{atm}(\tau)$ and $\frac{\partial}{\partial \tau}\omega_{atm}(\tau)$ for each $\tau \in \mathcal{T}_{aux}$ which is used during the training process.

5 Surface Stochastic Volatility Inspired (SSVI)

The following version of SSVI prior model (with power-law parameterization) has been used for the entire volatility surface.

$$\omega_{\text{ssvi}}^{\text{prior}}(k,\tau) = \frac{\omega_{\text{atm}}(\tau)}{2}\left(1 + \rho\phi k + \sqrt{(\phi(k+\rho)^2 + 1 - \rho^2}\right) \tag{SSVI}$$
$$\phi := \phi(\omega_{\text{atm}}(\tau)) = \frac{\eta}{\omega_{\text{atm}}(\tau)^\gamma (1+\omega_{\text{atm}}(\tau))^{1-\gamma}}$$

Here $\omega_{\text{atm}}(\tau)$ is the ATM term structure and ρ, γ and η are tunable parameters. In other words, in (Implied Variance), we have that

$$\theta_{prior} = (\rho, \gamma, \eta)$$

6 Convergence

Our convergence scheme utilizes periodic checkpoints, loss-based reinitialization, and weight perturbation to manage and improve the training process. See Table 3 for the techniques used to improve convergence.

Checkpoint Interval	A checkpoint is set every 500 epochs.
Bad Initialization	After the first 4 checkpoints, if the best loss is not below 1, the model is reinitialized.
Learning Rate Adjustment	Every 4 checkpoints, if the best loss is not below 0.05, the learning rate is reset to its initial value.
Weights Perturbation	After each checkpoint, regardless of other conditions, the weights of the model are perturbed. This is to help escape local minima.
Divergence Handling (Bad Perturbation)	If the current loss is at least 10% worse than the best loss so far and > 0.1, and this occurs after the first checkpoint, the models are reloaded from the last saved state, and training continues from the last checkpoint with the best loss value.

Table 3: Techniques to Improve Convergence

7 Results

We have two kinds of training: training a reference model and training models using the reference model for initialization. Training the reference model takes significantly more time, whereas training other models based on it takes much less time. Table 4 shows the parameter values of the reference model as well as an example of those of a subsequent model trained based on the reference model. As Figures 1 and 2 show, training the subsequent model is much faster.

Parameter	Reference Model	Subsequent Model (Ex)
α	0.7	0.5
β	-0.03	-0.04
κ	0.5	0.4
v_0	0.01	0.02
θ	0.0625	0.05
ρ	-0.75	-0.65
σ	1	0.8
λ	0.2	0.3

Table 4: Parameter Values

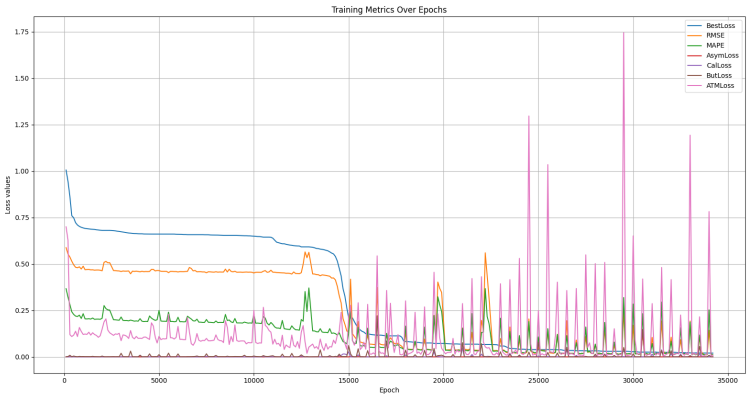

Figure 1: Training loss trajectory for the reference model, including tracking of various components such as RMSE, MAPE, and ATM loss. Refer to Table 4 for the parameter values.

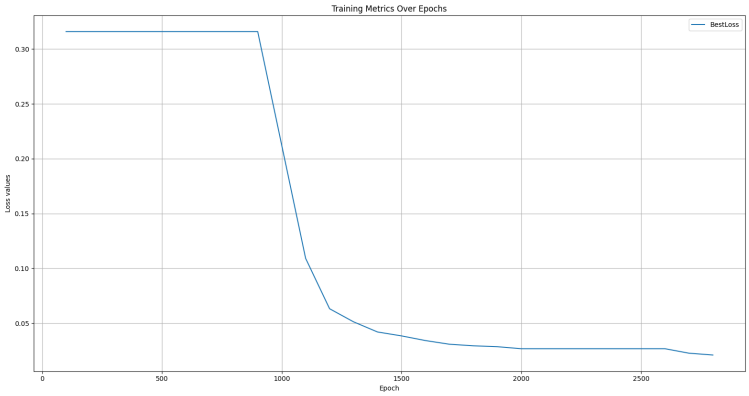

Figure 2: Training best loss trajectory for the subsequent model. See Table 4 for parameter values.

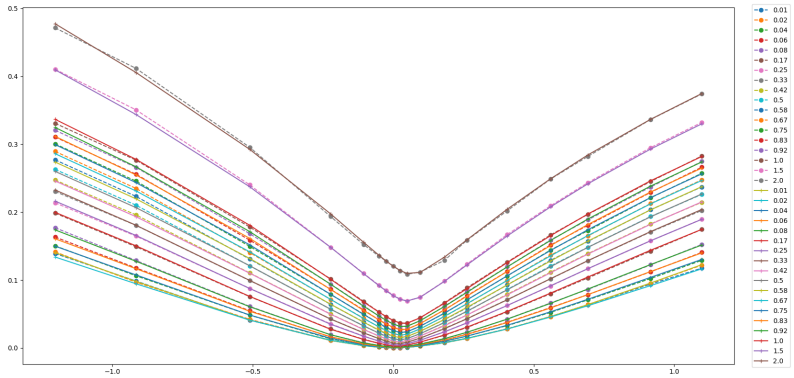

Figure 3: The reference model fits the implied variance surface perfectly, as indicated by the model output (- -), while ensuring no arbitrage opportunities across unobserved moneyness and maturities.

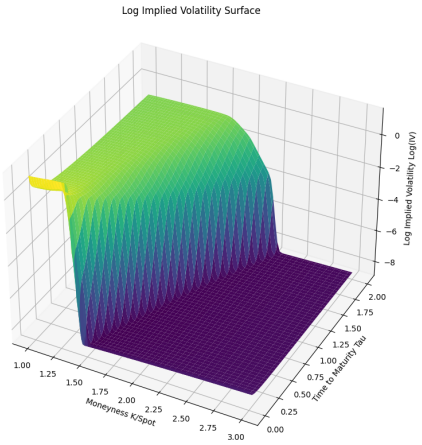

Figure 4: Log implied volatility surface generated by the reference model

8 Final Remarks

Several numerical experiments are still to be conducted using the project code. These experiments include applying deep smoothing to real market data rather

than synthetic data, performing backtesting, examining W-shaped smile patterns, investigating different sizes for neural network layers, and comparing results with a stand-alone prior (which can be easily implemented by setting the neural network's forward pass output to 1). We encourage interested readers to explore these avenues further.

References

[CM99] Peter Carr and Dilip Madan. "Option valuation using the fast Fourier transform". In: *Journal of computational finance* 2.4 (1999), pp. 61–73.

[Rop10] Michael Roper. "Arbitrage free implied volatility surfaces". In: *preprint* (2010).

[ATV20] Damien Ackerer, Natasa Tagasovska, and Thibault Vatter. "Deep smoothing of the implied volatility surface". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11552–11563.