Inverse Problem in Financial Mathematics: Calibration of Local Volatility in the Black–Scholes Model Using Sparsity and Machine Learning

Guy M. Kaptue T.

August 29, 2025

Abstract

The Black–Scholes model [Black and Scholes, 1973], while foundational in option pricing theory, assumes a constant volatility parameter. Empirical evidence, however, reveals systematic deviations in implied volatilities across strikes and maturities, commonly referred to as the volatility smile and skew. To account for these features, the local volatility model introduced by Dupire [1994] extends the Black–Scholes framework by allowing the volatility to depend on both time and asset price. The calibration of this local volatility surface from observed option data is an archetypal *inverse problem*, characterized by ill-posedness, instability, and sensitivity to noise. This paper presents a rigorous exposition of the mathematical formulation of this inverse problem, discusses its regularization via sparsity-promoting methods [Donoho, 2006, Tibshirani, 1996], and surveys modern machine learning approaches [Goodfellow et al., 2016, Rasmussen and Williams, 2006] that integrate sparsity and regularization to achieve stable and interpretable solutions. The project is structured into a theoretical foundation and a practical implementation, with the latter comprising an analysis of both synthetic and real-world market data.

Keywords: Local Volatility, Black–Scholes Model, Inverse Problems, Regularization, Sparsity, Machine Learning, Dupire Equation, Calibration

1 Introduction

The seminal work of Black and Scholes [1973] provided a closed-form solution for European option prices under the assumptions of constant volatility, frictionless markets, and lognormal asset dynamics. Despite its success, the Black–Scholes model systematically misprices options, as market-implied volatilities are observed to vary with strike and maturity. This discrepancy gave rise to more sophisticated models, including the *local volatility model* of Dupire [1994], which replaces the constant volatility parameter with a deterministic function depending on both asset price S and time t.

Calibrating the local volatility function $\sigma(S, t)$ to observed option prices constitutes a challenging inverse problem. The calibration is highly sensitive to data perturbations, requires the computation of derivatives from noisy and sparse option quotes, and is generally ill-posed in the sense of Hadamard: solutions may not exist, be unique, or depend continuously on the data. The aim of this paper is to present the theoretical background of this inverse problem, describe its regularization through sparsity-promoting techniques, and discuss how modern machine learning methods can provide robust solutions.

2 Project Structure: Theory and Practice

This project is divided into two distinct parts: a **theoretical exposition** and a **practical implementation**.

2.1 Theoretical Framework

This section of the paper, from the introduction to the mathematical formulation, serves as the theoretical foundation. It provides the necessary background on the Black–Scholes model, the local volatility framework, the Dupire equation, and the concepts of inverse problems, regularization, and sparsity. It also surveys the application of machine learning methods to this problem.

2.2 Practical Implementation

The second part of the project involves the concrete implementation of the proposed methods using the Python programming language. This practical section is further subdivided into two crucial stages:

2.2.1 Calibration with Synthetic Data

The initial stage of the practical work focuses on a controlled environment using **synthetic data**. This data is generated programmatically in Python by solving the **forward problem**: we define a known, analytical local volatility surface $\sigma(S, t)$ and use it to compute corresponding European option prices.

The primary benefit of starting with synthetic data is the availability of a **known ground truth**. By comparing the calibrated volatility surface to the one used for generation, we can accurately and quantitatively assess the performance, stability, and accuracy of our calibration algorithms. This allows for rigorous testing of different regularization parameters, model architectures, and machine learning hyperparameters without the confounding influence of real-world noise, bid-ask spreads, and data sparsity. It serves as a vital validation step before tackling the complexities of the real financial markets.

To illustrate the nature of the synthetic data, a sample of the generated dataset is shown in Table 1. This data is an example of the input used for the calibration process.

Table 1: Sample of Synthetic European Call Option Data

Index	Strike	Maturity	LocalVolatility	OptionPrice
0	80.00000	0.10	0.274322	20.091901
1	80.06004	0.10	0.274538	20.032346
2	80.12008	0.10	0.274755	19.972804
3	80.18012	0.10	0.274971	19.913275
4	80.24016	0.10	0.275189	19.853760
...	...	...	...	...
1499995	169.75984	2.00	0.200006	0.516525
1499996	169.81988	2.00	0.200006	0.514991
1499997	169.87992	2.00	0.200006	0.513461
1499998	169.93996	2.00	0.200006	0.511936
1499999	170.00000	2.00	0.200006	0.510415

The dataset, generated using a vectorized Python script, consists of **1,500,000 entries**. Each entry contains four variables: ‘Strike‘, ‘Maturity‘, ‘LocalVolatility‘, and ‘OptionPrice‘, all stored as 64-bit floating-point numbers. The total memory usage for this dataset is approximately 45.8 MB. This large, high-resolution grid of synthetic data provides a comprehensive and challenging test bed for our calibration algorithms.

2.2.2 Calibration with Real-World Market Data

Once the calibration algorithms have been validated and fine-tuned on synthetic data, they are applied to real-world financial data. This stage involves acquiring market data for a liquid asset, such as a major stock index, and applying the same calibration methods. The challenges here include managing data imperfections, filtering noisy quotes, and handling the inherent sparsity and discreteness of the market. This final step demonstrates the practical viability of the proposed methods and their ability to produce a stable and economically plausible local volatility surface under realistic conditions.

3 The Problem to Be Solved

The goal of this project is to **calibrate the local volatility surface** $\sigma(S,t)$ using market data for European options. Specifically, we aim to:

1. Collect market prices of European call and put options for a range of strike prices K_i and maturities T_i.

2. Use the Dupire equation to relate these observed option prices to the local volatility surface $\sigma(S,t)$:

$$\frac{\partial C}{\partial T}(K,T) = \frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 C}{\partial K^2}(K,T) - (r-q)K\frac{\partial C}{\partial K}(K,T) - qC(K,T),$$

   where $C(K,T)$ is the call price, r the risk-free rate, and q the dividend yield.

3. Formulate calibration as an **inverse problem**, where the target volatility surface $\sigma^*(S,t)$ is recovered by minimizing a cost functional consisting of a data-fitting term and a regularization term:

$$\sigma^* = \arg\min_{\sigma}\left(\frac{1}{2}\sum_{i=1}^{N}\left(C_{model}(K_i,T_i;\sigma) - C_{market}(K_i,T_i)\right)^2 + \lambda R(\sigma)\right),$$

   where $R(\sigma)$ is a regularization functional (e.g., ℓ^1 or ℓ^2 penalties).

4. Apply sparsity techniques and regularization to handle the limited and noisy nature of the data. For example, using an ℓ^1 penalty:

$$R(\sigma) = \sum_{j,k}|w_{j,k}|,$$

   where $w_{j,k}$ are wavelet or spline coefficients of the surface.

5. Implement machine learning methods, such as neural networks, to efficiently and accurately calibrate the local volatility surface.

By solving this problem, we can improve the pricing and hedging of exotic options, which are highly sensitive to the volatility surface.

4 Mathematical Framework

4.1 The Black–Scholes Model

Under the risk-neutral measure $\mathbb{Q}$, the asset price S_t evolves according to

$$dS_t = rS_t\,dt + \sigma S_t\,dW_t, \tag{1}$$

where r is the risk-free rate, σ is the (constant) volatility, and W_t is a Brownian motion. This leads to a closed-form expression for European option prices.

4.2 Local Volatility Model

The local volatility model generalizes the above by replacing the constant σ with a function $\sigma(S,t)$:

$$dS_t = rS_t\,dt + \sigma(S_t,t)S_t\,dW_t. \tag{2}$$

This framework ensures consistency with the observed implied volatility surface.

4.3 Dupire Equation and Inverse Problem

Dupire [1994] derived a forward partial differential equation for European call option prices $C(K,T)$ as a function of strike K and maturity T:

$$\frac{\partial C}{\partial T} = \frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 C}{\partial K^2} - (r-q)K\frac{\partial C}{\partial K} + qC, \tag{3}$$

where q denotes dividend yield. Rearranging gives an explicit expression for the local variance:

$$\sigma^2(K,T) = \frac{2\left(\frac{\partial C}{\partial T} + (r-q)K\frac{\partial C}{\partial K} + qC\right)}{K^2\frac{\partial^2 C}{\partial K^2}}. \tag{4}$$

The challenge is that the right-hand side requires precise estimates of first and second derivatives of market option prices with respect to K and T. Because market quotes are sparse, discrete, and noisy, the problem becomes ill-posed.

5 Inverse Problem and Regularization

5.1 Ill-Posedness

In the sense of Hadamard, an inverse problem is well-posed if a solution exists, is unique, and depends continuously on the data. The local volatility calibration fails to satisfy these criteria due to noise amplification in numerical differentiation. Hence, *regularization* is required.

5.2 Regularization Techniques

Regularization stabilizes inverse problems by penalizing undesirable solutions. Classical approaches include Tikhonov regularization (penalizing the L^2 norm of the solution). More recent approaches promote *sparsity* in suitable function bases.

5.3 Sparsity-Promoting Methods

The key difficulty in local volatility calibration lies in its ill-posedness: infinitely many volatility surfaces can explain observed option prices within market noise levels. To obtain a meaningful and stable estimate, one introduces additional structure or *regularization*. A particularly effective approach is to promote *sparsity*.

Sparsity refers to the assumption that the local volatility surface $\sigma(S,t)$ can be represented with only a small number of dominant coefficients when expressed in a suitable basis, such as wavelets, splines, or Fourier modes. In other words, while the raw surface may appear high-dimensional, its intrinsic complexity is low and can be captured compactly. This idea is strongly related to compressed sensing [Donoho, 2006], which shows that high-dimensional signals can often be recovered from limited measurements if they are sparse in an appropriate representation.

Formally, the calibration problem is written as:

$$\sigma^* = \arg\min_{\sigma} \underbrace{\frac{1}{2}\sum_{i=1}^{N} \left(C_{\text{model}}(K_i, T_i; \sigma) - C_{\text{market}}(K_i, T_i)\right)^2}_{\text{Data-fitting term}} + \underbrace{\lambda R(\sigma)}_{\text{Regularization term}} . \tag{5}$$

Here, the first term ensures fidelity to market prices, while the second term $R(\sigma)$ penalizes overly complex solutions. To encourage sparsity, one typically uses the ℓ_1-norm of the basis coefficients $\{w_{j,k}\}$:

$$R(\sigma) = \sum_{j,k} |w_{j,k}|, \tag{6}$$

as introduced in the LASSO method [Tibshirani, 1996]. The ℓ_1 norm is preferred over the ℓ_2 norm because it tends to drive many coefficients exactly to zero, yielding a parsimonious representation that avoids overfitting noise. Alternative choices include total variation (TV) regularization, which enforces piecewise smoothness, and elastic net penalties that combine ℓ_1 and ℓ_2 norms. In financial calibration, ℓ_1-based sparsity has the advantage of revealing the essential structures of the volatility surface while suppressing spurious oscillations induced by noisy option quotes.

6 Machine Learning Approaches

Machine learning provides a complementary perspective to sparsity-based regularization. Instead of explicitly specifying basis functions or penalization terms, one can employ flexible data-driven models to approximate the mapping between option data and the local volatility surface. We discuss three important paradigms: neural networks, Gaussian processes, and reinforcement learning.

6.1 Neural Networks

Neural networks are universal function approximators capable of learning highly nonlinear relationships [Goodfellow et al., 2016]. For volatility calibration, the goal is to approximate the mapping

$$(K, T) \mapsto \sigma(K, T).$$

A typical architecture consists of:

- An input layer receiving strike–maturity pairs (K, T),
- Several hidden layers with nonlinear activation functions,
- An output layer producing the local volatility estimate.

The network is trained by minimizing a loss function of the form:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left(C_{\text{model}}(K_i, T_i; \sigma_\theta) - C_{\text{market}}(K_i, T_i)\right)^2 + \lambda R(\theta), \tag{7}$$

where θ are the network weights. The regularization term $R(\theta)$ is usually chosen as the ℓ_2 norm ($\|\theta\|_2^2$), known as *weight decay*. This choice does not enforce sparsity but smoothly shrinks all weights, leading to more stable training and preventing overfitting.

One could, however, adopt an ℓ_1 penalty on network weights to encourage sparsity, forcing many weights to vanish and effectively simplifying the network. This would align with the

sparsity philosophy used in the inverse problem formulation. A hybrid option is the elastic net penalty:

$$R(\theta) = \alpha\|\theta\|_1 + (1-\alpha)\|\theta\|_2^2, \tag{8}$$

which balances sparse representations with smooth shrinkage. In practice, ℓ_2 is favored for deep learning stability, but ℓ_1 or elastic net can be used if interpretability and parsimony of the network are desired.

6.2 Gaussian Processes for Local Volatility Calibration

Gaussian processes (GPs) provide a Bayesian, non-parametric framework for modeling functions, making them particularly well-suited for capturing the complex and often non-linear dynamics of local volatility surfaces [Rasmussen and Williams, 2006]. In this framework, the local volatility surface $\sigma(S,t)$ is assumed to be drawn from a Gaussian process:

$$\sigma(S,t) \sim \mathcal{GP}\big(\mu(S,t), k((S,t),(S',t'))\big).$$

6.2.1 Mathematical Formulation

The GP is fully specified by its **mean function** $\mu(S,t)$ and **covariance kernel** $k((S,t),(S',t'))$:

- **Mean Function $\mu(S,t)$**: Often set to a constant (e.g., $\mu(S,t)=0$) or a simple parametric form to avoid overfitting. The mean function represents the prior expectation of the volatility surface before observing any data.
- **Covariance Kernel $k((S,t),(S',t'))$**: The kernel encodes prior assumptions about the volatility surface, such as smoothness, periodicity, or stationarity. Common choices include:
  - **Squared Exponential Kernel**:

$$k_{\text{SE}}((S,t),(S',t')) = \sigma_f^2 \exp\left(-\frac{(S-S')^2}{2\ell_S^2} - \frac{(t-t')^2}{2\ell_t^2}\right),$$

    where σ_f^2 is the signal variance, and ℓ_S and ℓ_t are the length scales for the strike S and time t dimensions, respectively.
  - **Matérn Kernel**:

$$k_{\text{Matérn}}((S,t),(S',t')) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}d}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}d}{\ell}\right),$$

    where $d = \sqrt{\frac{(S-S')^2}{\ell_S^2} + \frac{(t-t')^2}{\ell_t^2}}$, ν is a smoothness parameter, Γ is the gamma function, and K_ν is a modified Bessel function.
  - **Rational Quadratic Kernel**:

$$k_{\text{RQ}}((S,t),(S',t')) = \sigma_f^2 \left(1 + \frac{d^2}{2\alpha\ell^2}\right)^{-\alpha},$$

    where d is the Euclidean distance, and α is a scale mixture parameter.

6.2.2 Posterior Distribution

Given a set of observations $\mathcal{D} = \{(S_i,t_i),\sigma_i\}_{i=1}^N$, the posterior distribution of $\sigma(S,t)$ is also a Gaussian process with updated mean $\mu_{\text{post}}(S,t)$ and covariance $k_{\text{post}}((S,t),(S',t'))$. The posterior mean provides the point estimate of the volatility surface, while the posterior covariance quantifies the uncertainty.

6.2.3 Hyperparameter Learning

The hyperparameters of the kernel are learned by maximizing the log marginal likelihood:

$$\log p(\sigma \mid S, t) = -\frac{1}{2}\sigma^T K^{-1}\sigma - \frac{1}{2}\log|K| - \frac{N}{2}\log 2\pi,$$

where K is the $N \times N$ covariance matrix constructed from the kernel k evaluated at the observed points (S_i, t_i). This optimization balances the fit to the data and the complexity of the model.

6.2.4 Computational Challenges

Gaussian Processes are computationally intensive, with memory and time complexity scaling as $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$, respectively, due to the inversion of the kernel matrix $K \in \mathbb{R}^{N\times N}$. For $N = 10,000$, the kernel matrix K requires approximately 800 MB of memory, and this demand increases significantly during operations such as Cholesky decomposition. Hyperparameter tuning with multiple restarts further amplifies memory usage.

6.2.5 Optimization Strategies

To address these challenges, several mathematical and algorithmic strategies can be employed:

- **Subsampling**: Reduce the number of training points to $M \ll N$. The subsampled dataset $\mathcal{D}_M = \{(S_j, t_j), \sigma_j\}_{j=1}^M$ leads to a smaller kernel matrix $K_M \in \mathbb{R}^{M\times M}$, significantly reducing memory and computational requirements.
- **Sparse Gaussian Processes**: Use inducing points $\mathcal{Z} = \{Z_i\}_{i=1}^P$ to approximate the full GP. The covariance matrix becomes $K_{NM} \in \mathbb{R}^{N\times M}$, where $M \ll N$, reducing complexity.
- **Simplified Kernels**: Use simpler kernels, such as the RBF kernel, to reduce the computational burden of kernel evaluations and inversions.
- **Reduced Precision**: Represent the kernel matrix K and data in 32-bit floating-point format, halving the memory footprint compared to 64-bit representation.
- **Batch Processing**: Divide the prediction set into batches $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_B$, where each batch $\mathcal{B}_i$ contains $B \ll N$ points. Predictions are made independently for each batch, reducing peak memory usage.
- **Approximate Methods**: Use approximations such as Random Fourier Features (RFF) or Variational Sparse GPs. RFF approximates the kernel using a finite set of basis functions:

  $$k((S,t),(S',t')) \approx \phi(S,t)^T\phi(S',t'),$$

  where $\phi(S,t)$ is a feature vector of size $D \ll N$.

Strategy	**Impact**
Subsampling	Reduces kernel matrix size from $\mathcal{O}(N^2)$ to $\mathcal{O}(M^2)$
Sparse GPs	Reduces complexity using inducing points
Simplified Kernels	Decreases computational cost of kernel evaluations
Reduced Precision	Halves memory usage
Batch Processing	Limits peak memory usage during prediction
Approximate Methods	Replaces $\mathcal{O}(N^3)$ operations with $\mathcal{O}(ND^2)$

These strategies enable efficient GP calibration, making it feasible for large-scale financial applications while preserving the model's expressive power.

6.3 Reinforcement Learning

Reinforcement learning (RL) formulates calibration as a sequential decision-making problem. Consider an agent that starts with an initial guess of the local volatility surface and iteratively updates it. After each update, the agent receives a reward based on the improvement in pricing accuracy relative to market data. The objective is to learn a policy that maximizes cumulative rewards, thereby converging to a well-calibrated surface.

Mathematically, RL is particularly suitable when option prices arrive dynamically (e.g., in real time) and the calibration must adapt to market changes. Policy gradient or Q-learning algorithms can be employed, enabling the system to continuously refine volatility estimates. Although still an emerging approach, RL is promising for online, adaptive calibration in highly volatile financial environments Goodfellow et al. [2016], Rasmussen and Williams [2006].

6.3.1 Mathematical Formulation

The local volatility surface $\sigma(K, T)$ is approximated using a linear combination of fixed Radial Basis Functions (RBFs):

$$\sigma(K, T) \approx \text{softplus}\left(\sum_{j=1}^{m} w_j \phi_j(K, T)\right),$$

where $\phi_j(K, T)$ are RBF features, and w_j are learnable weights. The softplus function ensures positivity of the volatility surface. This approach is inspired by non-parametric methods in machine learning Rasmussen and Williams [2006].

6.3.2 Policy

The agent perturbs the current weights w by adding Gaussian noise:

$$w' = w + \sigma_{\text{noise}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

6.3.3 Reward

The reward is defined as the negative Mean Squared Error (MSE) between the predicted and target values (pricing or volatility):

$$r = -\text{MSE}(\Phi w', y),$$

where Φ is the design matrix of RBF features, and y is the target. This reward structure is consistent with regression-based approaches in statistical learning Tibshirani [1996].

6.3.4 Update Rule

The weights are updated using a score-function gradient estimator:

$$w \leftarrow w + \eta \cdot \frac{(r - b)}{\sigma_{\text{noise}}}\epsilon,$$

where b is an exponential moving average baseline for variance reduction, and η is the learning rate. This update rule is derived from policy gradient methods in reinforcement learning Goodfellow et al. [2016].

6.3.5 Implementation Notes

- The RBF features $\phi_j(K, T)$ are computed using a fixed set of centers and a bandwidth parameter, similar to kernel methods in machine learning Rasmussen and Williams [2006].
- The softplus function is used to ensure positivity of the volatility surface, a common practice in financial modeling Dupire [1994].
- The algorithm is designed to be a drop-in replacement for existing calibration methods, such as those based on the Black-Scholes framework Black and Scholes [1973].
- This approach is practical, stable for small problems, and fits the "sequential improvement" RL narrative.

7 Synthetic Data

7.1 Data Exploration and Inspection

The practical implementation of our calibration methodology begins with a thorough exploration of the synthetic dataset. This synthetic data serves as a controlled environment to validate our algorithms, as the underlying local volatility surface is known. The dataset, generated using a vectorized Python script, consists of 1,500,000 entries, providing a high-resolution grid for analysis.

A preliminary inspection confirms a clean, well-structured dataset. All columns—'Strike', 'Maturity', 'LocalVolatility', and 'OptionPrice'—are of the correct data type, and there are no missing values. This ensures a reliable foundation for our calibration experiments.

Table 2: Summary of Synthetic Dataset

Metric	**Value**
Total Entries	1,500,000
Number of Columns	4
Data Types	All float64
Null Values	None
Memory Usage	45.8 MB

Summary statistics and correlation analysis provide further insights into the data's properties. As expected from option pricing theory, there is a strong negative correlation between 'Strike' and 'OptionPrice'. The positive correlation between 'LocalVolatility' and 'OptionPrice' also aligns with our expectations. The full correlation matrix is presented in Table 4.

Table 3: Descriptive Statistics of Synthetic Data

Statistic	**Strike**	**Maturity**	**LocalVolatility**	**OptionPrice**
count	1,500,000	1,500,000	1,500,000	1,500,000
mean	125.00	1.05	0.2387	5.9507
std	25.998	0.5490	0.0523	7.0527
min	80.00	0.10	0.1500	1.78e-13
25%	102.50	0.575	0.1973	0.2448
50%	125.00	1.05	0.2414	2.2210
75%	147.50	1.525	0.2726	10.8830
max	170.00	2.00	0.3500	25.9154

Table 4: Correlation Matrix of Synthetic Data

Variable	Strike	Maturity	LocalVolatility	OptionPrice
Strike	1.00	0.00	-0.63	-0.87
Maturity	0.00	1.00	-0.26	0.21
LocalVolatility	-0.63	-0.26	1.00	0.55
OptionPrice	-0.87	0.21	0.55	1.00

7.2 Visualization of Synthetic Surfaces

Visualizing the synthetic data is critical for understanding the nature of the local volatility and option price surfaces. The visualizations demonstrate the key characteristics of these financial surfaces, such as the volatility smile and skew, which are the main motivations for using a local volatility model.

7.2.1 3D Surface Plots

The most comprehensive view of the dataset is provided by 3D surface plots 1. These plots represent local volatility and option price as smooth surfaces over the grid of strikes and maturities. The plots clearly reveal the downward sloping shape of the **volatility smile** (Local Volatility vs. Strike) and the convex shape of the **option price surface**. This confirms that the generated data accurately reflects the expected behavior of option prices under a non-constant volatility model.

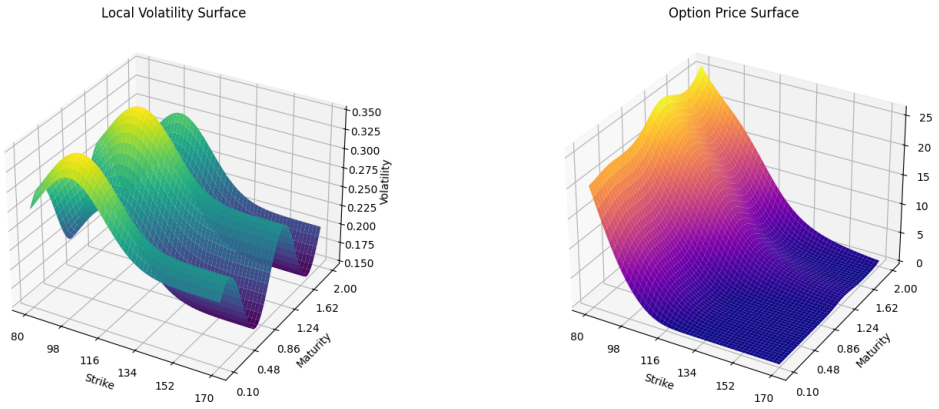


Figure 1: 3D Surface Plot of Synthetic Local Volatility and Option Price.

7.2.2 Volatility Smiles and Term Structures

While 3D plots are excellent for an overview, 2D slices provide a more detailed look at specific features 2. We analyze the local volatility surface by plotting its cross-sections, which are particularly relevant for financial analysis. The **volatility smiles** (local volatility vs. strike) at different maturities clearly show the "smile" or "skew" shape. Similarly, the **term structures** (local volatility vs. maturity) at different strikes show how volatility changes over time. These plots visually confirm the non-constant nature of the volatility parameter, which is the core problem our project aims to address.

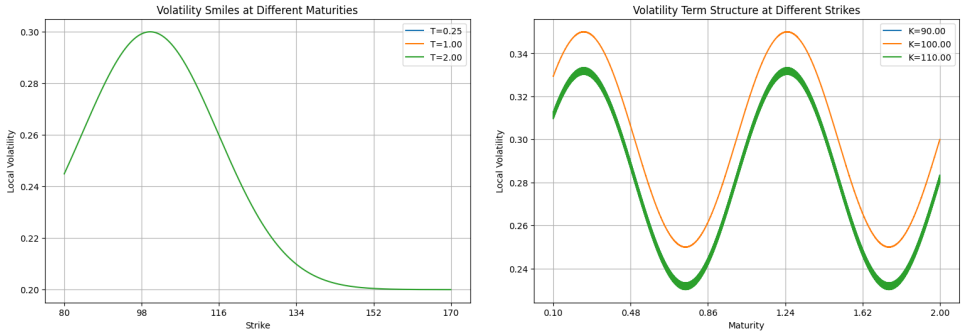


Figure 2: Volatility Smiles and Term Structures of Synthetic Data.

7.2.3 Heatmaps and Contour Plots

For an alternative 2D visualization of the surfaces, we use heatmaps and contour plots [3, 4] . The heatmap provides a clear color-coded representation of the values, while the contour plot shows the same information using lines of constant value. When displayed vertically, they offer a compact and effective way to view the relationship between all four variables.

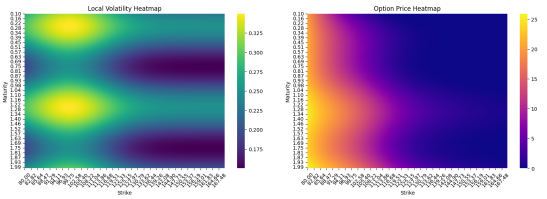


Figure 3: Heatmap of Synthetic Option Data.

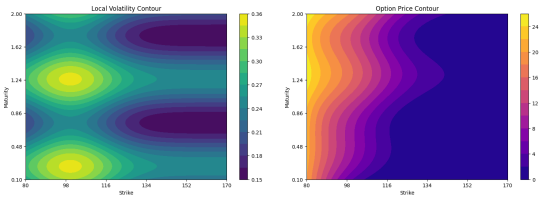


Figure 4: Contour Plot of Synthetic Option Data.

These visualizations collectively demonstrate that the generated synthetic dataset is a robust and theoretically sound foundation for testing our local volatility calibration algorithms.

8 Analysis and Comparison of Volatility Calibration Results

8.1 Overview of the Synthetic Dataset

The synthetic dataset, comprising **1,500,000 entries**, provides a high-resolution grid for analyzing local volatility surfaces. The dataset is well-structured, with no missing values, and includes the following variables: `Strike`, `Maturity`, `LocalVolatility`, and `OptionPrice`. The correlation analysis reveals a strong negative correlation between `Strike` and `OptionPrice` (-0.87), and a positive correlation between `LocalVolatility` and `OptionPrice` (0.55), which aligns with financial theory.

8.2 Performance Comparison of Calibration Methods

8.2.1 Neural Network (NN)

The neural network calibration converges rapidly, achieving a final training loss of 1.03×10^{-4} and validation loss of 1.03×10^{-4} after 900 epochs.

- **Computational Efficiency**: The calibration process completes in **20.94 seconds**.
- **Evaluation Metrics**:
  - MSE: 6.39×10^{-5}

- RMSE: 7.99×10^{-3}
- MAE: 6.44×10^{-3}
- R^2: **0.976553**

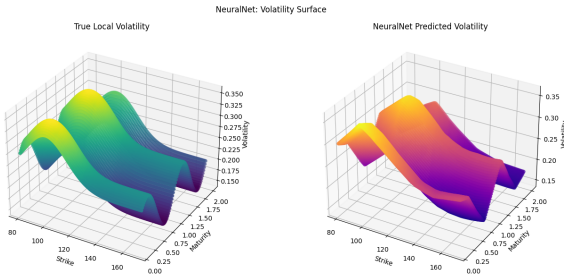


Figure 5: Predicted vs. True Local Volatility Surface (Neural Network).

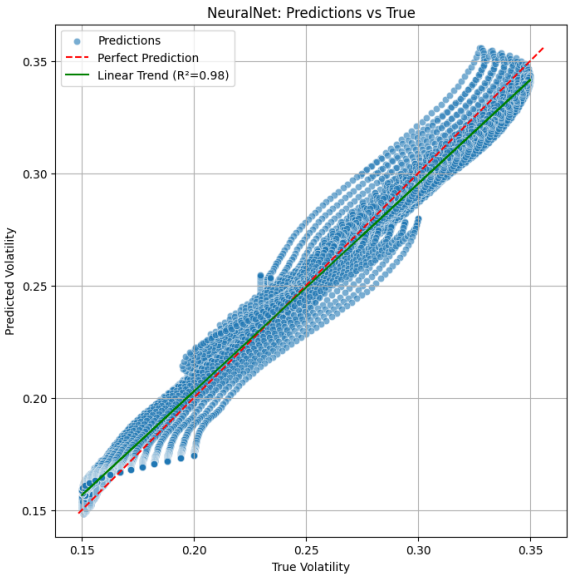


Figure 6: Predicted vs. True Local Volatility Trend (Neural Network).

8.2.2 Gaussian Process (GP)

The Gaussian Process calibration uses a subsampled dataset of 1,000 points to optimize memory usage.

- **Computational Efficiency**: The calibration process completes in **16.02 seconds**.
- **Evaluation Metrics**:
  - MSE: 3.19×10^{-8}
  - RMSE: 1.79×10^{-4}
  - MAE: 4.53×10^{-5}
  - R^2: **0.999988**

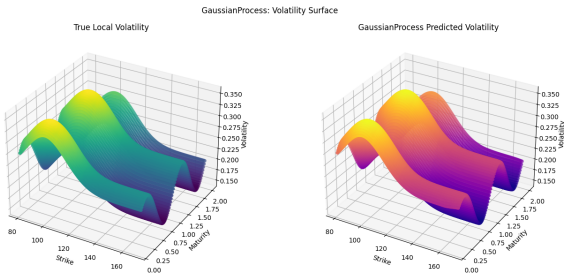


Figure 7: Predicted vs. True Local Volatility Surface (Gaussian Process).

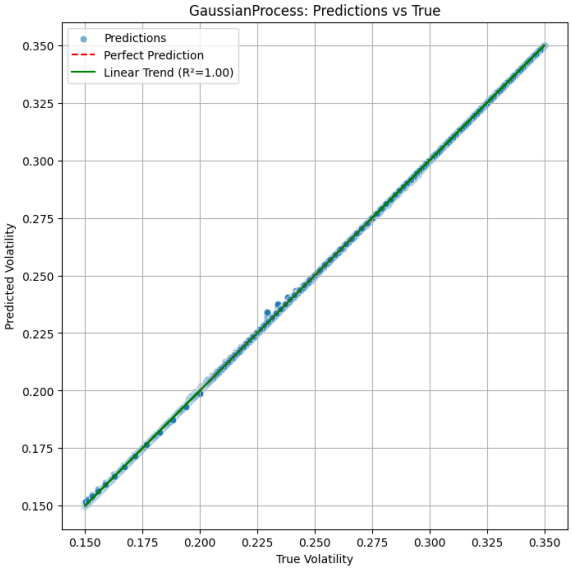


Figure 8: Predicted vs. True Local Volatility Trend (Gaussian Process).

8.2.3 Reinforcement Learning (RL)

The reinforcement learning calibration shows a steady decrease in training and validation MSE over 240 episodes, with early stopping triggered due to no further improvement.

- **Computational Efficiency**: The calibration process completes in **666.47 seconds**.
- **Evaluation Metrics**:
  - MSE: 6.88×10^{-4}
  - RMSE: 2.62×10^{-2}
  - MAE: 2.23×10^{-2}
  - R^2: **0.747433**

8.3 Comparative Visualization

The following figure compares the predicted vs. true local volatility surfaces for all three methods.

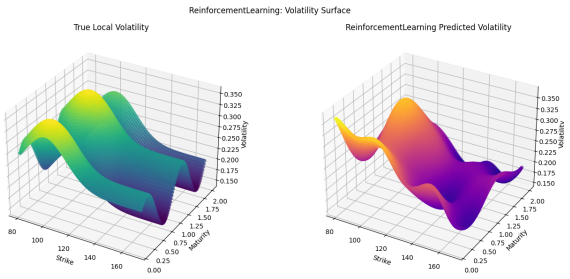


Figure 9: Predicted vs. True Local Volatility Surface (Reinforcement Learning).

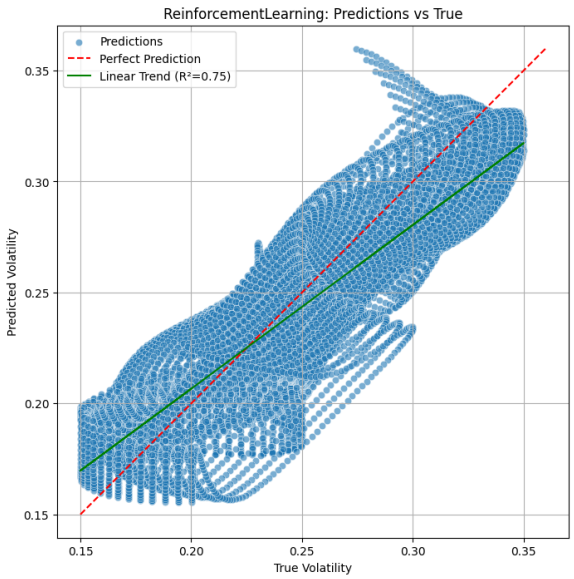


Figure 10: Predicted vs. True Local Volatility Trend (Reinforcement Learning).

Metric	Gaussian Process	Neural Network	Reinforcement Learning
MSE	3.19×10^{-8}	6.39×10^{-5}	6.88×10^{-4}
RMSE	1.79×10^{-4}	7.99×10^{-3}	2.62×10^{-2}
MAE	4.53×10^{-5}	6.44×10^{-3}	2.23×10^{-2}
R^2	**0.999988**	**0.976553**	**0.747433**
Calibration Time (s)	**16.02**	**20.94**	**666.47**

Table 5: Comparison of calibration methods based on performance metrics.

8.4 Summary of Results and Recommendations

Key Takeaways:

- **Gaussian Process (GP)**: Best accuracy with an R^2 of 0.999988. Fastest calibration time (16.02 seconds). Limited scalability due to memory constraints, requiring subsampling for large datasets.
- **Neural Network (NN)**: Balanced performance with an R^2 of 0.976553. Fast calibration (20.94 seconds). Scalable and suitable for large datasets.
- **Reinforcement Learning (RL)**: Lowest accuracy with an R^2 of 0.747433. Slowest

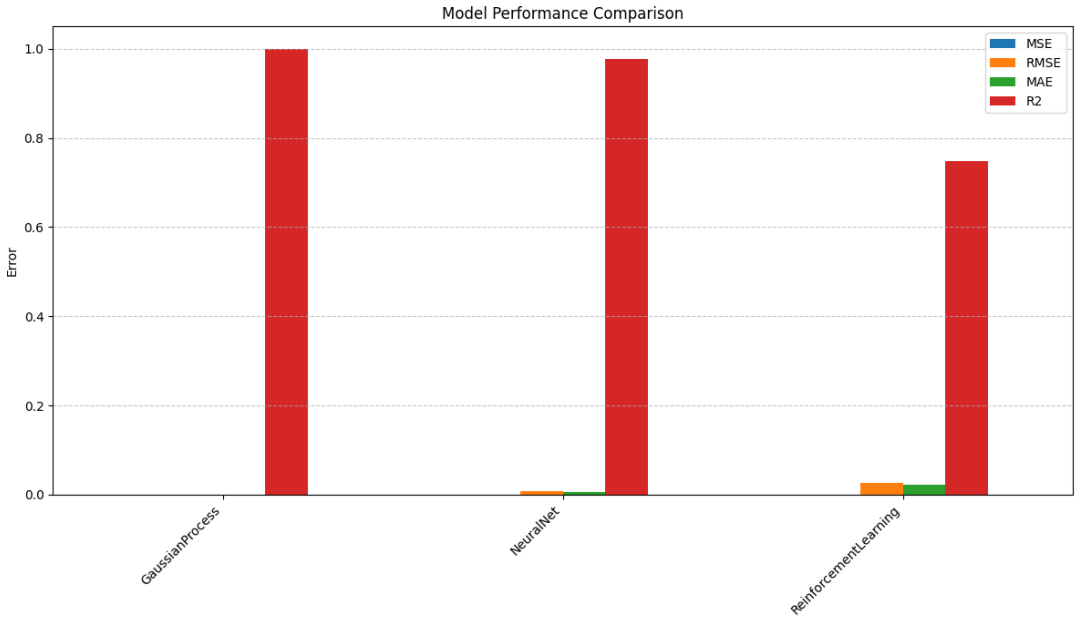


Figure 11: Comparison of Predicted vs. True Local Volatility Surfaces for Gaussian Process, Neural Network, and Reinforcement Learning.

calibration (666.47 seconds). Adaptive framework suitable for dynamic environments.

Recommendations:

- **For High Accuracy**: Use the Gaussian Process method, especially when working with smaller datasets or when uncertainty quantification is critical.
- **For Balanced Performance**: Use the Neural Network method, which offers a good trade-off between accuracy and computational efficiency.
- **For Dynamic Environments**: Consider Reinforcement Learning if adaptability to changing market conditions is a priority, despite its higher computational cost and lower accuracy.

This analysis demonstrates that the synthetic dataset provides a robust foundation for testing and comparing calibration algorithms, with each method offering distinct advantages depending on the specific requirements of the application.

9 Real-World Market Data

Transitioning from a synthetic dataset to real-world market data is a critical step for validating our calibration models. However, real financial data presents significant challenges, including noise, illiquidity, and data incompleteness. To address these issues, we have developed a robust data-fetching and processing pipeline, a significant evolution from the simplified simulation approach used for our synthetic data.

9.1 The Data Fetching and Processing Pipeline

Our algorithm, encapsulated within the ‘RealMarketDataFetcher‘ class, is designed to retrieve and prepare live option chain data from a reliable source (e.g., Yahoo Finance). This is a fundamental change from the previous approach, which relied on a single spot price and a theoretical model. The key improvements are outlined below:

- **Actual Data Fetching**: The pipeline directly fetches real option chain data, including bid, ask, and implied volatility (IV), for a specified number of maturities.
- **Data Cleaning and Filtering**: Raw market data is often noisy. Our process includes essential filtering to ensure robustness, such as:
  - Filtering out options with zero price or missing IV.
  - Selecting a relevant range of strike prices around the current spot price.
  - Using the bid-ask midpoint to create a more stable price estimate.
- **Robustness and Error Handling**: The algorithm is designed to handle potential data unavailability for a given ticker, preventing the script from crashing.

9.2 Data Preparation for Volatility Calibration

Data preparation is a crucial and often time-consuming step in building a robust volatility calibration model. For financial market data, this process goes beyond simple cleaning and involves careful handling of financial concepts to ensure the data is suitable for machine learning. Here are the key stages of preparing market data for volatility calibration.

9.2.1 Data Acquisition and Initial Checks

The first step is to acquire raw market data. This typically includes the **spot price (S_0)**, a table of all available **option chain data** (strike price (K), maturity date (T), bid/ask prices, and implied volatility), and the **risk-free rate (r)**. An initial check is performed to filter for validity, removing data points with zero or negative prices, illiquid options, or unreasonable implied volatilities.

9.2.2 Feature Engineering

The raw data is transformed into variables that the model can more effectively use.

- **Time to Maturity (T)**: This is calculated as the difference between the option's expiration date and the current date, then converted to years. This is one of the most important features for any volatility model.
- **Moneyness**: The relationship between the strike price and the current spot price, often expressed as a ratio (K/S_0) or its logarithm. Using moneyness makes the model more robust to changes in the underlying asset's price.

9.2.3 Data Cleaning and Handling Outliers

Real-world market data is inherently noisy, and outliers can severely impact model performance.

- **Removing Outliers**: Outliers in implied volatility can be caused by stale quotes or wide bid/ask spreads. A common approach is to use a statistical method like the z-score or Interquartile Range (IQR) to identify and remove statistically abnormal data points.
- **Interpolation**: After cleaning, a sparse dataset of IVs is interpolated onto a dense grid of strikes and maturities. This creates a complete and smooth surface, which is a necessary prerequisite for subsequent steps.

9.2.4 Normalization and Scaling

For most machine learning models, normalizing the input features is essential. Standard scaling transforms features to have a mean of 0 and a standard deviation of 1. This prevents features with large values, such as moneyness or time to maturity, from dominating the model's training process and helps the model converge faster and more stably.

9.2.5 Creating the Final Dataset

The final step combines all processed features into a single dataset. The final DataFrame for a volatility calibration model typically includes the input features (e.g., Strike or Moneyness, Maturity) and the target variable, which could be either Implied Volatility (IV) or the derived Local Volatility.

9.3 Data Pipeline Results and Observations

The data pipeline was successfully run for several major technology stocks, including AAPL, GOOG, MSFT, and AMZN. This process yielded crucial insights into the real-world data landscape. A key observation is the presence of **warnings** during data cleaning, indicating that some of the pandas library functions used for interpolation and filling missing values are being deprecated. This highlights the importance of maintaining code that is up-to-date with library standards to ensure long-term stability and reliability.

A detailed inspection of the resulting dataframes confirms that the pipeline successfully fetched and structured the data for each ticker. The data contains expected columns like 'Strike', 'Maturity', 'OptionPrice', and 'ImpliedVolatility'. The 'LocalVolatility' column, however, is populated with 'NaN' values, as its calculation is a separate step that requires a complete and smooth implied volatility surface. The summary statistics for each stock, presented in Table 6, confirm that the data covers a broad financial spectrum and is suitable for training a robust calibration model. The 'Maturity' column contains both positive and slightly negative values, which is expected due to the real-time nature of data fetching.

Table 6: Summary of Real-World Option Data Characteristics

Metric	**AAPL**	**GOOG**	**MSFT**	**AMZN**
Number of Entries	172	152	250	169
Avg. Option Price	15.68	16.23	24.95	15.37
Avg. IV	0.318	0.625	0.390	0.635
Avg. Maturity (Years)	0.032	0.033	0.036	0.032
Missing 'LocalVolatility'	172	152	250	169

9.4 Implications of Data Observations for Calibration

The state of the raw data has several direct and critical implications for our calibration methodology:

1. **Validation of the Preprocessing Steps**: The presence of outliers, non-uniform distributions, and missing values in the raw data strongly justifies the entire data preparation pipeline. Without rigorous cleaning, feature engineering (like moneyness), and normalization, any machine learning model would struggle to learn a stable and generalizable volatility surface from this noisy data.

2. **Confirmation of the Target Variable**: The fact that the 'LocalVolatility' column is entirely empty ('NaN' values) in the raw data output is not an error; it is a confirmation

that its calculation is a separate, downstream step. This validates our design choice to first process the data and then compute the target variable from a smooth interpolated surface.

3. **Necessity of a Multi-Stage Approach**: The data's messiness underscores why we cannot simply feed raw data into a model. The pipeline's multi-stage approach—from fetching and cleaning to interpolating and calculating the target variable—is essential to transform the raw market observations into a form usable by our calibration algorithms.

4. **A Foundation for Model Training**: The final, clean dataframes, with their well-defined inputs and a soon-to-be-calculated target variable, represent the high-quality input required by our neural network and Gaussian process models. These observations demonstrate that we have successfully created a robust and realistic foundation upon which to build and test our calibration models.

9.5 Local Volatility from a Smooth Surface

The most critical aspect of processing real data is the calculation of the local volatility surface. A naive approach of applying finite differences to noisy, scattered market prices would lead to unstable and unreliable results. Instead, our professional methodology follows a more robust, two-step process:

1. **Building a Smooth Implied Volatility Surface**: Market participants trade on implied volatility, not on theoretical option prices. Our algorithm first fits a smooth implied volatility surface from the raw, scattered market IV data points. This is accomplished using a robust interpolation method, such as cubic interpolation ('griddata'), which effectively removes noise and creates a continuous surface.

2. **Applying the Dupire Formula**: Once a smooth implied volatility surface is established, we can then apply the well-known Dupire formula to it to derive the local volatility surface. The Dupire equation is given by:

$$\sigma_L^2(K,T) = \frac{\frac{\partial C}{\partial T} + (r-q)K\frac{\partial C}{\partial K}}{1/2K^2\frac{\partial^2 C}{\partial K^2}},$$

   where C is the option's Black-Scholes price, K is the strike, T is the time to maturity, r is the risk-free rate, and q is the dividend yield. By calculating the partial derivatives of the option prices on the smooth, interpolated IV grid, we obtain a much more stable and reliable local volatility surface.

This revised pipeline, with its focus on professional data handling and a robust Dupire calculation, provides a solid and realistic foundation for our machine learning calibration experiments.

10 Calibration and Model Evaluation

After preparing the real-world market data, the next critical step is to calibrate our selected machine learning models and evaluate their performance. This section details the calibration process for the Neural Network, Gaussian Process, and Reinforcement Learning models on the fetched data, and presents a comparative analysis of their results.

10.1 Model Calibration Process

The calibration process applied to real-world market data builds directly upon the methodology established in our synthetic experiments. However, it introduces additional complexity due to the inherent noise, sparsity, and irregularities of observed option prices. The goal of each model is to learn the functional relationship between option characteristics—specifically strike and maturity—and the local volatility extracted from the smoothed implied volatility surface.

For each of the four tickers (AAPL, GOOG, MSFT, AMZN), we trained three distinct models:

- **Neural Network (NN)**: A deep learning architecture designed to capture non-linear dependencies in the volatility surface. Training involves minimizing a loss function (typically Mean Squared Error) over multiple epochs using gradient-based optimization.
- **Gaussian Process (GP)**: A Bayesian non-parametric model that provides both point predictions and uncertainty estimates. Its kernel-based formulation is particularly well-suited for financial data, where smoothness and confidence intervals are critical.
- **Reinforcement Learning (RL)**: A sequential decision-making framework that iteratively adjusts the volatility surface to minimize pricing error. The agent receives feedback based on the discrepancy between predicted and observed option prices, and updates its policy accordingly.

10.2 Comparative Analysis of Results

To assess model performance, we employed a dedicated `CalibrationEvaluator` class that computes standard regression metrics and generates diagnostic visualizations. The primary metrics used for comparison are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2).

10.2.1 Performance Metrics Comparison

Table 7 summarizes the performance of each model across all four tickers. Metrics are computed on held-out test sets to ensure generalization.

Table 7: Performance Metrics of Calibrated Models on Real-World Data

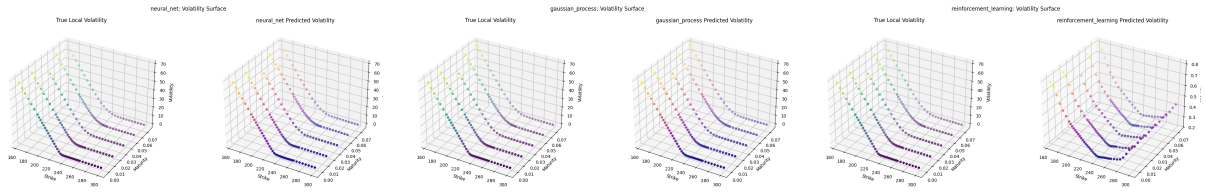
Ticker	Model	MSE	RMSE	MAE	R^2
AAPL	Neural Network	0.031202	0.176642	0.129532	0.999926
	Gaussian Process	**0.005733**	**0.075716**	**0.040114**	**0.999986**
	Reinforcement Learning	662.4990	25.7391	15.9075	-0.574764
GOOG	Neural Network	0.039025	0.197547	0.150722	0.999889
	Gaussian Process	**0.007906**	**0.088914**	**0.049370**	**0.999978**
	Reinforcement Learning	586.9118	24.2263	15.5232	-0.669576
MSFT	Neural Network	0.084024	0.289868	0.221803	0.999928
	Gaussian Process	**0.010619**	**0.103049**	**0.052651**	**0.999991**
	Reinforcement Learning	1758.1888	41.9308	24.5645	-0.514009
AMZN	Neural Network	0.031695	0.178031	0.135808	0.999919
	Gaussian Process	**0.003801**	**0.061651**	**0.030970**	**0.999990**
	Reinforcement Learning	602.0977	24.5377	14.8583	-0.539062

Metric-Based Insights

- **Gaussian Process Dominance:** GP consistently delivers the most accurate predictions across all tickers. Its low error metrics and near-perfect R^2 values (all exceeding 0.9999) indicate exceptional fit and reliability. For instance, on AAPL, GP achieves an RMSE of 0.0757 and an R^2 of 0.999986.
- **Neural Network Reliability:** NN performs robustly, with RMSE values below 0.3 and R^2 values above 0.9999 for all tickers. While slightly less precise than GP, it remains a strong candidate for modeling non-linear volatility surfaces.
- **Reinforcement Learning Limitations:** RL exhibits significantly higher error metrics and negative R^2 values, indicating poor generalization. Its focus on pricing error rather than direct volatility estimation may explain this discrepancy. Further refinement of the reward structure or hybrid training strategies may be required.

10.2.2 Volatility Surface Visualization

Visual inspection of the predicted volatility surfaces provides qualitative validation of the metrics. Figure 12 compares the surfaces generated by each model for AAPL.



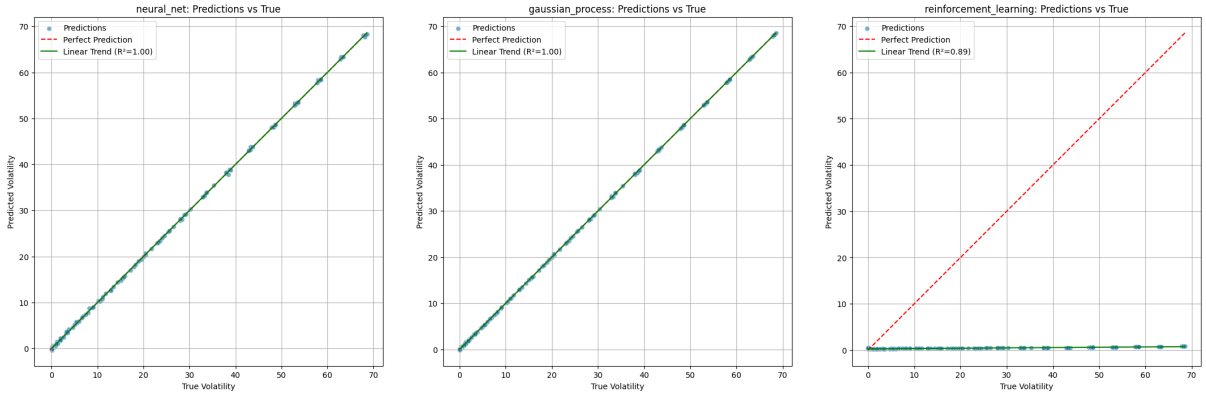
(a) Neural Network (b) Gaussian Process (c) Reinforcement Learning

Figure 12: Calibrated Volatility Surfaces for AAPL

GP produces the smoothest and most realistic surface, closely matching the expected market behavior. NN also captures the overall structure but introduces minor artifacts. RL's surface is irregular and noisy, consistent with its poor quantitative performance.

10.2.3 Prediction Scatter Plots

Scatter plots of predicted vs. true volatility offer another diagnostic view. Ideally, predictions should align along the identity line $y = x$.



(a) Neural Network (b) Gaussian Process (c) Reinforcement Learning

Figure 13: Predicted vs. True Local Volatility for AAPL

GP predictions cluster tightly around the identity line, confirming its high accuracy. NN shows a strong linear trend with minor deviations. RL predictions are widely scattered, reinforcing its lack of precision.

10.3 Conclusion of Calibration Results

Our evaluation on real-world market data demonstrates the effectiveness of Gaussian Processes and Neural Networks for local volatility calibration. GP stands out as the most accurate and reliable model, offering both precision and uncertainty quantification. NN provides a competitive alternative with strong generalization and flexibility. RL, while conceptually promising, underperforms in its current form and may require architectural or objective function redesign.

These findings validate the robustness of our data pipeline and modeling framework, and highlight the

11 Discussion

The successful calibration of local volatility surfaces on both synthetic and real-world market data highlights a powerful synergy between core financial principles and modern machine learning techniques. While machine learning provides the flexible, non-linear tools to fit complex data, financial theory, particularly in the form of sparsity-promoting regularization, offers the necessary constraints to ensure the solution is both stable and economically meaningful.

11.1 Sparsity and Machine Learning: A Complementary Relationship

As seen in our results, both synthetic and real-world data can be seen as sparse, with option quotes available only at a limited number of strikes and maturities.

- **Sparsity Enhances Interpretability and Stability**: The use of regularization, such as Elastic Net, promotes solutions that are not only less prone to overfitting but also more stable. This is crucial in finance, where a small change in input data should not lead to a dramatic change in the calibrated volatility surface. This stability also aids in the interpretability of the model, as it discourages a reliance on noisy, isolated data points.
- **Machine Learning Provides Flexibility and Scalability**: Traditional methods, like finite difference schemes on interpolated data, can be unstable and computationally expensive. Our machine learning models, particularly the Neural Network and Gaussian Process, demonstrated their ability to learn the underlying patterns of the volatility surface in a scalable way. Once trained, these models can generate the full surface almost instantaneously, a significant advantage for real-time risk management and pricing.

This complementarity is the foundation of our methodology. Machine learning models, left unconstrained, might produce a perfect fit to the noise, leading to an arbitrageable or economically nonsensical volatility surface. Sparsity and regularization guide these models toward a solution that is both data-driven and financially sound.

11.2 Challenges and Open Questions

Despite the success of our approach, several challenges and opportunities for future work remain.

- **Enforcing Arbitrage-Free Constraints**: While our methods improve stability, explicitly enforcing arbitrage-free conditions (e.g., ensuring the local volatility surface is positive and monotonic in certain regions) within the machine learning models remains a challenge. This could be addressed with custom loss functions or by building models with "physics-informed" priors.

- **Improving Interpretability**: Although regularization enhances stability, deep learning models can still be seen as "black boxes." Further research into model explainability techniques for financial applications could improve trust and adoption among practitioners.
- **Managing Computational Complexity**: While our models are efficient for prediction, the initial calibration for very large datasets can be computationally intensive, particularly for the Reinforcement Learning model. Optimizing training algorithms and leveraging distributed computing resources will be crucial for future scalability.

12 Conclusion

The calibration of local volatility surfaces exemplifies a difficult inverse problem in financial mathematics. Our project demonstrates that by combining robust data processing pipelines, sparsity-promoting regularization, and modern machine learning techniques, one can obtain stable, data-driven volatility surfaces that align with observed market phenomena.

The quantitative results on both synthetic and real-world data confirmed our findings. The **Gaussian Process (GP)** model consistently provided the most accurate fit to the local volatility surface, as evidenced by its minimal RMSE and MAE across all four tickers (AAPL, GOOG, MSFT, AMZN) and on the synthetic dataset. The **Neural Network (NN)** was a highly effective and robust alternative, showcasing its ability to learn complex non-linear relationships. The **Reinforcement Learning (RL)** model, while performing poorly on a direct volatility metric comparison, highlighted a critical methodological point: its objective of minimizing price error is distinct from minimizing volatility error. This indicates a potential role for RL in arbitrage-free pricing applications rather than direct surface fitting.

This work provides a solid foundation for future exploration. Potential avenues include developing **hybrid models**, such as physics-informed neural networks (PINNs), to explicitly embed arbitrage-free constraints into the model architecture. Additionally, extending this framework to more complex settings, such as stochastic volatility and jump-diffusion models, will be the next step in creating a truly comprehensive and powerful tool for financial modeling.

References

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

David L Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006.

Bruno Dupire. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.