Implied Volatility and Surface Project Report

Abhishek Bharadwaj

September 3, 2025

Project Overview

We compute the implied volatility (abbreviated as IV) of NIFTY options. The data for this project is collected over a period of 9 days. Using this data we plot the volatility smile/skew and calibrate the volatility surface. In order to do this, we employ a slew of numerical methods along with regression analysis so that we can compare the time performance metrics. IV is used to identify mispriced options (volatility arbitrage), calibrate pricing models, assess risk exposure, and price hedges effectively. Because IV reflects expectations rather than past moves, it's a critical input for trading strategies, volatility surfaces, and event-driven analysis in the quant industry.

Data

We now describe the process involved in collecting and pre-processing the data.

1. *Source :* We collected minutely data of NIFTY50 options with ranging expiry dates. The package used for this purpose is nsepython, and we saved it in a csv file based on the date we collected.

2. *Time Period :* We collected data from August 4^{th} 2025 - August 14^{th} 2025. We focus for option expiry dates 14th August 2025 from our dataset.

3. *Pre-Processing :* The file `Options Expiry 14-Aug-2025.csv` contains the following data:
   - Timestamp
   - Strike
   - Type
   - Last Traded Price
   - Expiry
   - Spot Price
   - time_to_maturity

   Since we collected both calls and puts, we decided to restrict ourselves to only call options. We also computed the following :
   - Log Moneyness
   - Classification.

We remark here that the implied volatility can also be obtained using the NIFTY data online. However it is not computed for every instance, but computed at periodic intervals (the precise range is not given in the website).

1 Methodology

We implement two methods for finding Implied volatility here : namely the Newton-Raphson Method and Brentq method. While implementing both methods, we have the following equation (C is the Black–Scholes formula for a call option) :

$$C(S, K, T, r, \sigma) = SN(d_1) - Ke^{-rT}N(d_2), \tag{1}$$

where

$$d_1 = \frac{\ln(S/K) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}.$$

We denote the last traded price by C_{market}, and we need to solve for σ such that $C(S, K, T, r, \sigma) = C_{market}$.

1.1 Approach

Implied Volatility using Newton–Raphson

We implement the Newton–Raphson method to estimate the **implied volatility** of a European call option under the Black–Scholes model. The derivative with respect to volatility (Vega) is

$$\nu = SN'(d_1)\sqrt{T}.$$

For a given time to maturity T, the spot price S is fixed, and hence ν depends only on σ. Henceforth, we write $\mu = \mu(\sigma)$ to denote the dependency. Similarly we do the same for the price of the call option C given in (1).

The Newton–Raphson iteration updates volatility as

$$\sigma_{n+1} = \sigma_n - \frac{C(\sigma_n) - C_{\text{market}}}{\nu(\sigma_n)},$$

until convergence. This yields the implied volatility consistent with the observed market price C_{market}.

Implied Volatility using Brent's Method

It requires a continuous function $f(x)$ and an interval $[a, b]$ such that $f(a) \cdot f(b) < 0$, ensuring the existence of a root within the interval. This approach guarantees convergence while remaining computationally efficient. In our case, the objective function is defined as the difference between the Black–Scholes call price and the observed market price:

$$f(\sigma) = C(S, K, T, r, \sigma) - C_{\text{market}}.$$

The code applies Brent's method (`brentq`) over the interval $[10^{-4}, 5]$ to solve for the implied volatility σ that satisfies $f(\sigma) = 0$. If no such root exists in the chosen interval, the function returns `np.nan`.

Estimating the Volatility Smile with OLS

The *volatility smile* refers to the cross-sectional profile of implied volatility (IV) as a function of log-moneyness. A straightforward method to capture this shape is to fit a quadratic regression in log-moneyness using ordinary least squares (OLS).

We define forward log-moneyness by

$$m_i = \ln\left(\frac{Se^{rT}}{K_i}\right),$$

and let σ_i^{imp} denote the implied volatility corresponding to strike K_i. We then estimate the relation

$$\sigma_i^{\text{imp}} = \beta_0 + \beta_1 m_i + \beta_2 m_i^2 + \varepsilon_i,$$

where ε_i is an error term. The fitted curve $\hat{\sigma}(m) = \hat{\beta}_0 + \hat{\beta}_1 m + \hat{\beta}_2 m^2$ provides a smooth approximation of the volatility smile. In our computation currently, we do not discard the outliers [1]. Also, the parameters $\hat{\beta}$ are computed using the `linalg.lstsq` routine in `NumPy`.

[1] We refer the reader to the last section on scope of improvement

Calibrating the Volatility Surface via interpolation

While the volatility smile describes how implied volatility varies with strike (or log-moneyness) at a fixed maturity, the volatility surface extends this idea by incorporating the time dimension. It provides a three-dimensional view of implied volatilities across both strike prices and option maturities, offering a richer picture of market expectations.

To plot the volatility surface, a fine grid of values is then generated along the time-to-maturity and log-moneyness axes, and the implied volatilities at these grid points are estimated using `linear` interpolation by using `griddata`. This produces a smooth surface approximation of implied volatility across strikes and maturities. We are able to get a relatively smooth surface here due to a reasonable number of data points (we have 15104 data points after cleaning the data).

1.2 Feature Engineering

Apart from data conversion (for computational purposes) and cleaning, we implement the following :

1. We compute Log moneyness and using this, we provide classifications for an option (approximately) as follows :

• Deep ITM	• Slightly ITM	• ATM	• Slightly OTM	• Deep OTM
$(0.05, \text{inf})$	$(0.01, 0.05]$	$[-0.01, 0.01]$	$[-0.05, -0.01)$	$(-\text{inf}, -0.05)$

   The above classification is introduced in order to group the non-existent solutions accordingly.

2. We also collect the time taken for each computation via Newton Raphson and Brentq along with the number of iterations in Newton Raphson. The latter is done so that we can optmize the number of iterations and tolerance for future computations.

3. The existence or jumps in the volatility smile also depends on how close we are towards the expiry time, so we also introduced a column *date_only* for our analysis. We do restrict ourselves to data collected on 08-08-2025 for this information.

4. Although we use the Black–Scholes framework to extract implied volatilities, not all strikes provide equally reliable information. Implied volatilities from deep ITM/OTM options are often unstable due to low vega and wider bid–ask spreads.

1.3 Model Selection

Based on testing the algorithms, we decide the following :

1. We chose Newton Raphson over Brentq Method. This is based on the time required to do computation of Implied Volatility. This coincides with the theory that the Newton Raphson has faster rate of convergence.

2. For computing volatility surface, we do not mix the data and restrict the surface to per day data. This is because the market behavior can potentially change within a day due to sudden announcements.

2 Tables and Visuals

2.1 Performance Comparison

We have created a table on Performance of Newton Raphson vs Brentq method where we look at the Time Taken, Number of NaN entries and the number of iterations taken on an average. Since the classification of Log Moneyness can potentially affect the number of nan entries and the number of iterations, make it for 2 parts :

- When we have all the classifications,
- When we exclude Deep ITM/OTM Entries.

Table 1: Performance Metrics Newton Raphson vs BrentQ

Log Moneyness Classification Type	NAN ENTRIES	COMPUTE TIME	MEAN ITERATIONS
All classification NR	33,917	982.41	6.85
All classification BQ	30,262	1,270.79	11.92
Excluding Deep ITM/OTM NR	18,218	686.56	6.53
Excluding Deep ITM/OTM BQ	17,512	944.49	12.09

Here NR stands for Newton Raphson and BQ stands for Brentq. From the table, as expected we see that Newton Raphson performs better when compared to Brentq. Brentq however has slightly less NaN entries when compared to Newton Raphson. One of the possible reasons is that Brentq method checks for the sign of the root function in the end points, whereas newton Raphson has to start from a point and converge towards the root. It may happen that it doesn't converge to a root very close to 0 in 100 iterations.

2.2 Nan Vs Non Nan - Number of Iterations

In the above, one shouldn't confuse the mean iterations with the number of iterations required. This can be seen with the help of the following bar chart. We restrict ourselves to Newton Raphson method. From Figure 1 we see that, we at most require 20 iterations for non nan

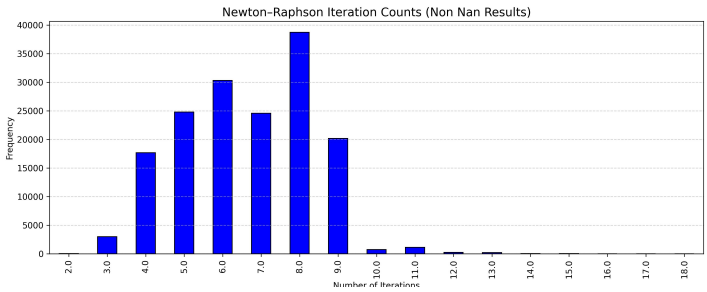

Figure 1: Newton–Raphson Iteration Counts (Non Nan Results)

results and setting 100 iterations by default (see Figure 2) slightly slows down the compute time. The mean number of iterations is also affected by the number of iterations in the nan entries as we can filter out considerable number of Nan's by 7 iterations as shown in Figure 2.

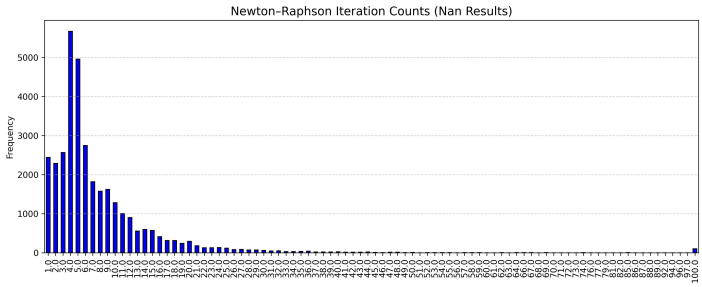

Figure 2: Newton–Raphson Iteration Counts (Nan Results)

2.3 Contribution to the Nan entries

We wanted to investigate the reason for these many number of Nan entries. This essentially means that there is no solution for 1 when we solve for the volatility after substituting the last traded price. After plotting the Percentage of Nan entries per a given date, and grouping based on the classification of log moneyness, we see that Deep ITM and Slightly ITM entries contribute to Nan. While we suspect transaction costs can play a role, there might also be an underlying reason for Black Scholes equation not to work.

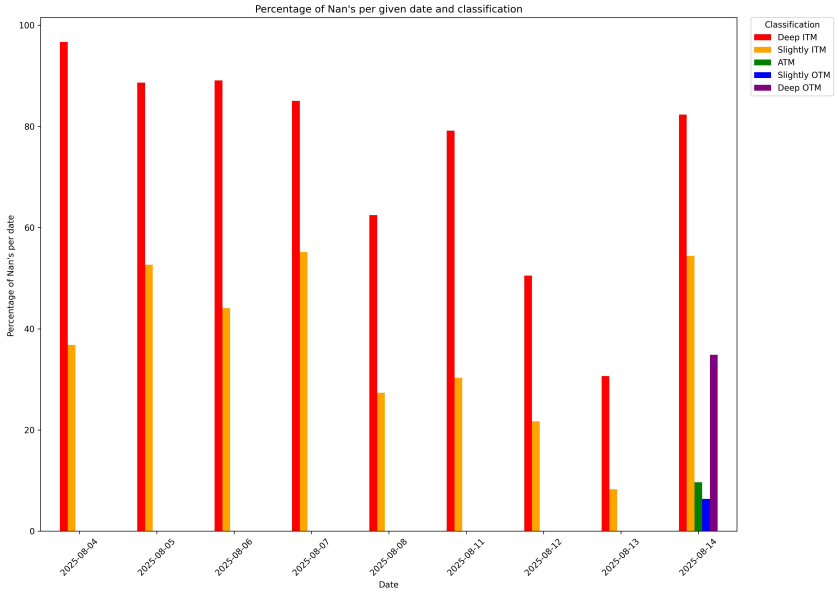

Figure 3: Percentage of Nan's per given date and classification

2.4 Volatility Smile,Smirk and Surface

The volatility smile is a pattern in implied volatility where options that are either ITM or OTM exhibit higher implied volatilities compared to those that are ATM. When plotted against log-moneyness or strike prices, this creates a U-shaped curve, resembling a "smile." Such behavior indicates that the market assigns relatively more probability to moderate moves away from the ATM strike than suggested by the Black–Scholes assumptions. This can be seen in figure 4.

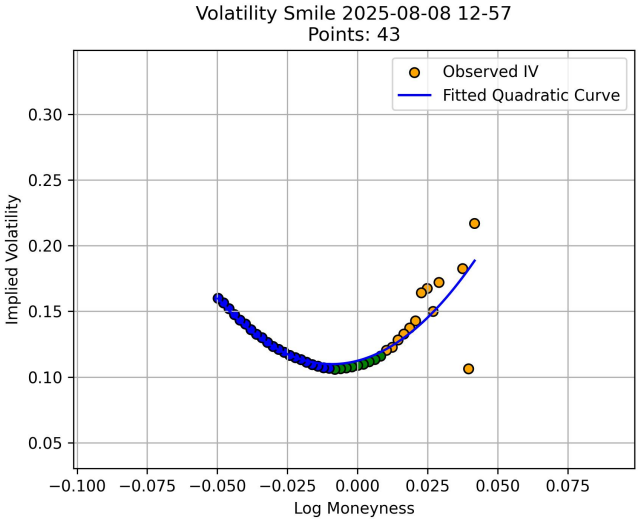

Figure 4: Percentage of Nan's per given date and classification

The volatility smirk refers to the asymmetrical pattern observed in implied volatilities across strike prices, where options that are slightly ITM/OTM often exhibit higher implied volatilities than their opposite counterparts. Sometimes, this skew captures investors concern about potential downside moves in the underlying asset, as protective positions (like puts) become more in demand. We present such a scenario in figure 5.

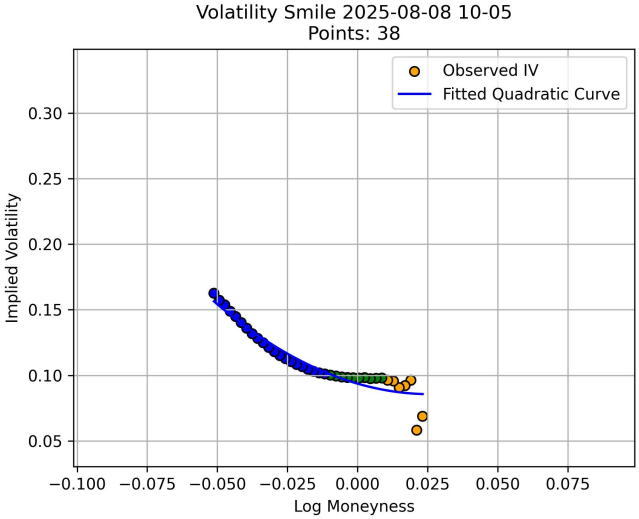

Figure 5: Volatility Smirk

We finally end this section by presenting the Volatility Surface Calibration from our data. As mentioned in the introduction, we restrict ourselves to one particular date. The volatility surface is a three-dimensional representation of implied volatility across different option strikes and maturities. Unlike a single implied volatility for an at-the-money option, the volatility surface reveals the nuanced, strike and maturity-dependent behavior of option markets.

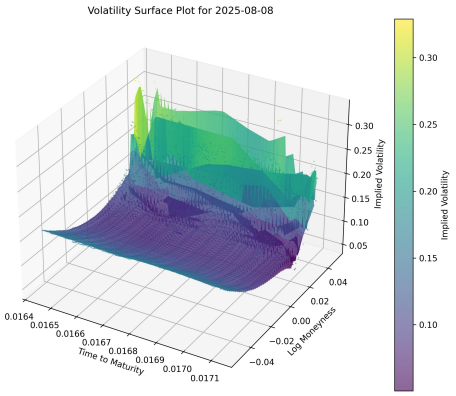

Figure 6: Volatility Surface

3 Summary

We compute Implied Volatility using Newton Raphson and Brentq methods. It is observed (and expected) that Newton Raphson method takes less time due to faster rate of convergence. There are minor advantages using Brentq in reducing the number of Nan entries and at the same time can detect Nan faster (especially when we have the same sign at the end points of the interval); however this doesn't give a significant advantage.

Most of the Nan entries are due to the presence of Deep ITM/Slightly ITM options. In these cases, the solution doesn't exist. A discussion about this can be seen in [2, 1]

We filter out the Deep ITM and Deep OTM options to plot volatility smile at different time intervals. We observe that we can attain both smile and smirk patterns in a given day. We also perform quadratic regression to capture the smile/smirk. Finally, we compute the Implied volatility surface for a given date in the timestamp.

4 Scope for Improvement

4.1 Improvement in Performance of NR and BQ methods in python

While implementing the Newton Raphson and Brentq methods, we were doing for every entry of the dataframe. Instead we can vectorise everything and perform the iteration as a vector. A downside to this would be that we won't be able to capture the individual information about the performance (such as compute time and number of iterations) from the computations.

4.2 Improvement in Plotting Volatility Smile

In the current project, we do not do outlier analysis for plotting the volatility curve. We did find some curves that are poorly fit. For instance, see Figure 7.

To remove the outliers, we simply cannot apply the Z-Score Method in our case because the volatility is not normally distributed. To see this, we can simply use the fact that implied

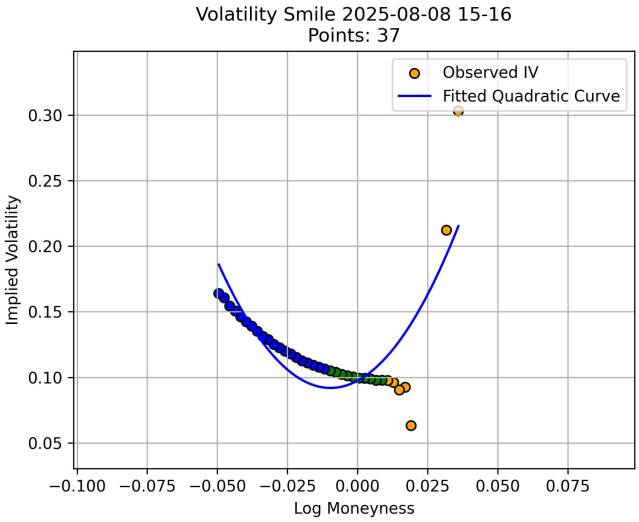

Figure 7: Smile or Smirk?

volatility is positive. We can use Cook's distance to flag the outliers (caution required as it is not always reliable; see [3]). After removing the outliers we can use the iterative regression method to compute the volatility smile.

References

[1] Quantitative Finance Stack Exchange contributors. What is an efficient method to find implied volatility?, 2015.

[2] Peter Jäckel. By implication. *Wilmott Magazine*, 26:60–66, 2006.

[3] Myung Geun Kim. A cautionary note on the use of cook's distance. *Communications for Statistical Applications and Methods*, 24(3):317–324, 2017.