Volatility Smile

Caleb Kilonzi

September 2024

Report

This Python code is designed to compute and visualize the implied volatilities from options data using the Black-Scholes model. It starts by reading an Excel file containing strike prices and call option prices for different time-to-maturity values (denoted as "tao"). Using the Newton-Raphson method, it calculates the implied volatilities for these options based on their market prices. The results are visualized in 3D plots, both as a rough surface and as a smoothed, interpolated surface. These plots provide insights into how implied volatility varies with strike prices and maturities.

Reading and Preprocessing Data

The code begins by reading an Excel file that contains options data. The file is assumed to have headers, with the second row being skipped. The first column is renamed 'Strike', and columns representing different maturities are renamed accordingly (e.g., tao=0.25, tao=0.5, etc.). The script also handles any unnecessary columns or rows, ensuring that the relevant data is prepared for further processing.

Once the data is loaded, a list of dictionaries (data_list) is created, storing each option's strike price, maturity (tao), and call option price. This structure is easy to loop through for subsequent implied volatility calculations.

Black-Scholes Pricing and Vega Calculation

The Black-Scholes formula is used to calculate the theoretical price of a European call option, given the current stock price (S), strike price (K), time to maturity (T), risk-free interest rate (r), volatility (sigma), and dividend yield (q). The function black_scholes_call() implements this formula, which is fundamental in determining the theoretical price of options.

In addition, the code defines a function black_scholes_vega() to compute the sensitivity of the option price to changes in volatility, known as Vega. This sensitivity is important for the Newton-Raphson method, as it helps adjust the volatility estimate when determining the implied volatility.

Newton-Raphson Method for Implied Volatility

Implied volatility is the volatility input that, when used in the Black-Scholes model, makes the theoretical option price equal to the market price. The Newton-Raphson method is employed to iteratively estimate this implied volatility. In the code, the function newton_raphson_iv() is designed to calculate implied volatility for each option by performing a single iteration of the Newton-Raphson method.

The method starts with an initial guess for volatility (initial_sigma = 0.15) and updates this estimate using the difference between the theoretical and market prices, adjusted by Vega. This approach allows the code to estimate the implied volatility for each option in the dataset.

Implied Volatility Calculation and Filtering

For each option in the dataset, the implied volatility is computed using the Newton-Raphson method. The results are stored in a list of dictionaries (implied_volatility_data), each containing the strike price, maturity, and the calculated implied volatility. After computing the implied volatilities, the code filters out any unreasonable values (i.e., implied volatilities less than 0 or greater than 3), which ensures that only meaningful results are used for visualization.

3D Visualization of Implied Volatility

The filtered implied volatilities are then plotted in 3D using matplotlib. The first plot is created using plot_trisurf(), which generates a triangulated surface where the axes represent strike prices, maturities, and implied volatilities. This provides an initial view of the implied volatility surface. The resulting plot is saved as implied_volatility_surface.pdf and displays the raw volatility data across different strike prices and maturities.

Interpolation and Smoothed Surface Plot

To provide a smoother and more continuous view of the implied volatility surface, the code interpolates the data using griddata(). This method creates a grid of strike prices and maturities and estimates implied volatilities at these grid points using cubic interpolation. The interpolated surface is then plotted using plot_surface(), resulting in a much smoother visualization of the volatility surface. This second plot is saved as implied_volatility_surface_interpolated.pdf and offers a clearer representation of how volatility behaves across different options.

Observation & Conclusion

The implied volatility surface is a critical tool in financial analysis, particularly in options pricing and risk management. It provides insights into how the market perceives volatility for different strike prices and maturities.

In the first plot, the implied volatility surface may appear rough, reflecting the data without any smoothing. This gives an accurate but somewhat noisy picture of market volatility. The

second, interpolated surface is smoother, making it easier to identify trends and patterns, though it assumes volatility changes smoothly, which may not always be the case in real markets.

The Newton-Raphson method used for calculating implied volatility is generally effective, but its success depends on the initial volatility guess and the nature of the option data. If the initial guess is too far from the actual implied volatility, or if the option price is far from what the Black-Scholes model predicts, the method may not converge within a single iteration. Multiple iterations or more refined methods may be needed for greater accuracy.

In conclusion, the code successfully computes implied volatilities from market data and visualizes them, offering valuable insights into market expectations of future volatility. These insights are crucial for option traders and risk managers, allowing them to price options more accurately and manage their portfolios effectively.