Implied Volatility Surface Generator using Black Scholes Model and Brent’s Method for Numerical Analysis

https://theivsurface.streamlit.app/

Micah Yihun Aldrich

University of Illinois Urbana Champaign

Abstract

Breaking into quantitative finance has become increasingly competitive in recent years. With the rise of distributed computing systems and the growing number of technically skilled students entering formal education, applications for quantitative finance roles have surged. Major financial institutions like JPMorgan now accept less than 1% of intern applications, training participants in cutting-edge technologies. Similarly, highly competitive trading firms like Jane Street offer internships with compensation exceeding $64,000 for 11 weeks, further driving up the number of applicants each year.

Understanding the intensity of this market, I aim to position myself as a valuable asset to banks, firms, and exchanges, ensuring a strong return on investment through my skills and research focus. My research interests lie in low-latency technologies and options theory, a field that combines advanced mathematics with creative problem-solving. This project focuses on developing a solution that demonstrates both my technical expertise and my innovative approach to options theory, specifically in areas like implied volatility.

The project is a web application that generates an implied volatility surface for a given set of parameters and stock options by symbol (e.g., SPY, NVDA). Built using Python, the Black-Scholes model, and Brent's method for the numerical analysis required to calculate implied volatility, the application is deployed on Streamlit. It showcases my proficiency in full-stack development and data visualization. Future iterations of the project will explore how it can be applied to develop arbitrage strategies for high-frequency trading firms.

Introduction

To appreciate the value of visualizing implied volatility as a 3D surface, it's essential to first understand options as financial derivatives, implied volatility, and their interrelationship.

At a high level, options are derivatives—financial contracts that derive their value from an underlying asset. An options contract gives the holder, also known as the buyer, the right, but not the obligation, to buy (call) or sell (put) the underlying asset at a specified price before a predetermined expiration date. The holder of a call option benefits from an increase in the underlying asset's value, while the holder of a put option benefits from a decrease in the asset's value. As such, the pricing of these contracts and the factors influencing that price (known as the premium) are of critical importance to option holders.

The premium of an options contract is composed of two key elements: intrinsic value and time value (also known as extrinsic value). The intrinsic value is the difference between the current price of the underlying asset and the strike price (the agreed-upon buy/sell price). The time value reflects the amount of time remaining until the contract expires and is equal to the premium minus the intrinsic value.

Several factors affect the premium of an options contract, including the underlying asset, strike price, time to expiration, dividends, interest rates, and volatility. Volatility refers to the fluctuation in the market price of the underlying asset and helps gauge the speed and magnitude of price movements. Implied volatility (IV) is a real-time estimate that predicts the future volatility of an asset. It provides a forecast of the underlying asset's price movement over the entire lifespan of the option.

An IV surface is a 3D plot that illustrates how implied volatility varies across different strike prices and expiration dates for options on the same underlying asset. By analyzing these surfaces, traders and investors can gain valuable insights into market expectations and uncover potential arbitrage opportunities.

Methodology

Ticker Data:
The ticker data used to calculate the options prices was sourced from *yfinance* (formerly known as fix-yahoo-finance), a popular open-source Python library that

retrieves financial data from the free Yahoo Finance API. *yfinance* was developed by Ran Aroussi and is widely used due to Yahoo Finance's broad selection of market data on stocks, bonds, currencies, and cryptocurrencies. Although Yahoo Finance once provided an official API, it was discontinued due to excessive misuse. As a result, alternatives like *yfinance* have become the primary choice for developers.

Tools & Libraries:
For this project, I utilized several open-source tools and libraries:

- Streamlit**:** A Python framework used to develop the front-end web application.
- Plotly: An interactive graphing library employed to visualize the Implied Volatility (IV) data.
- SciPy: A free and open-source Python library used for the numerical analysis required to calculate IV based on the Black-Scholes options pricing model, specifically leveraging SciPy's Brent's Method.

Brent's Method + Black-Scholes Model:
The backend of this project calculates the theoretical price of an options derivative using the Black-Scholes Model, which considers the following parameters:

- **S:** Current price of the underlying asset.
- **K:** Strike price of the option.
- **T:** Time to expiration (in years).
- **r:** Risk-free interest rate.
- **sigma:** Volatility (implied volatility, the value we aim to calculate).
- **q:** Dividend yield (default is 0).

The calculate_iv() function determines the implied volatility (sigma) that makes the theoretical Black-Scholes Model price match the actual market price of the option. This function takes the actual market price and the relevant Black-Scholes parameters as inputs.

The objective_function(sigma) calculates the difference between the actual market price of the option and the theoretical price produced by the Black-Scholes Model for a given volatility (sigma).

To find the correct value of sigma, I used Brent’s Method (brentq), a numerical root-finding algorithm that efficiently locates where a function crosses zero. Brent’s Method combines bisection, secant, and inverse quadratic interpolation methods. In this context, the goal is to find the implied volatility (sigma) where the difference between the theoretical Black-Scholes price and the actual market price equals zero.

Plotting the IV Surface:
The 3D surface plots represent the Implied Volatility (IV) across varying parameters. The x-axis denotes time to expiration, the y-axis represents strike price or moneyness (strike price / underlying asset price), and the z-axis shows the implied volatility for each option.

To create this visualization, I used the numpy.meshgrid function to generate a grid of points (T, K) for the X (time to expiration) and Y (strike price or moneyness) axes. To ensure the surface was visually appealing and smooth, I applied the griddata function, which interpolates the implied volatility (Z) values over the grid points, filling in gaps between known values with estimated ones.

For the final 3D surface plot, I used Plotly's go.Surface() to display the interpolated volatility values as a smooth surface. The color scale (I selected Plotly’s "Viridis") visually represents the volatility levels, offering an intuitive way to see how implied volatility changes across strike prices and time to expiration.

Implementation

Web Application Design:
The Streamlit interface allows users to interact with key model parameters such as the risk-free rate, dividend yield, and strike price filters through an intuitive sidebar. Users can input a ticker symbol, adjust parameters for option chains, and select visualization preferences (e.g., using strike price or moneyness for the y-axis). A nice feature of streamlit is that it dynamically updates the implied

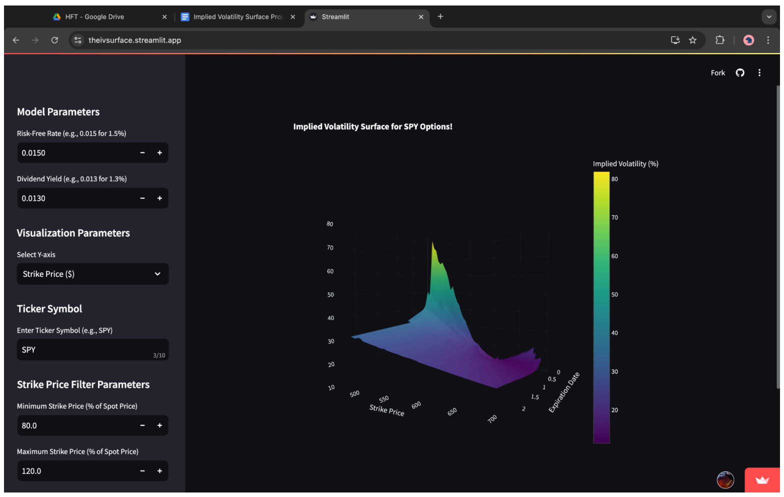
volatility surface based on the chosen parameters, offering a real-time, interactive experience for visualizing results.

Backend Architecture:
The backend retrieves live option chain data from Yahoo Finance (yfinance) and processes the data to calculate implied volatility using the Black-Scholes model and Brent's Method (Brentq) root-finding. The 3D surface plot is then updated in real-time using Plotly based on user inputs. Some challenges I had included ensuring smooth handling of large datasets for multiple expiration dates, maintaining real-time interactivity, and optimizing calculations for implied volatility across various strike prices and maturities. Future improvements would include; better implied volatility calculations, displaying the change in implied volatility surface for ticker over time and a more intuitive user interface for mobile devices.

Results

As of October 24, 2024, the result of this project is a front-end web application where users can intuitively input parameters via a sidebar on the left and visualize a smooth Implied Volatility (IV) Surface on the right side of the screen. The tool enables users to input any combination of parameters or results and back out missing values. The IV surface calculates the volatility implied by the market based on current call option prices for any configurable ticker symbol entered into the application.

The final output is a 3D surface, where the z-axis represents implied volatility (visualized through color), the y-axis represents strike price or moneyness, and the x-axis represents time to expiration. Each "slice" of this 3D surface can be viewed as a 2D plot for a given set of options, displaying the relationship between strike price and implied volatility. With additional inputs like the risk-free rate and dividend yield, the tool calculates call option prices across a range of strikes and expirations—up to about a year—using the Black-Scholes formula. The application then backs out the implied volatility based on the specified option prices. (as seen below)



theivsurface.streamlit.app

Model Parameters

Risk-Free Rate (e.g., 0.015 for 1.5%): 0.0150

Dividend Yield (e.g., 0.013 for 1.3%): 0.0130

Visualization Parameters

Select Y-axis: Strike Price ($)

Ticker Symbol

Enter Ticker Symbol (e.g., SPY): SPY

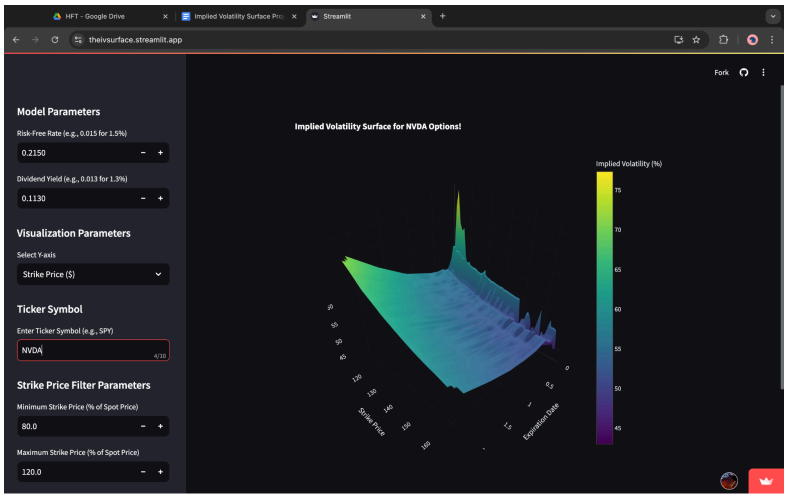
Strike Price Filter Parameters

Minimum Strike Price (% of Spot Price): 80.0

Maximum Strike Price (% of Spot Price): 120.0

Implied Volatility Surface for SPY Options!

Display with Ticker: SPY



theivsurface.streamlit.app

Model Parameters

Risk-Free Rate (e.g., 0.015 for 1.5%): 0.2150

Dividend Yield (e.g., 0.013 for 1.3%): 0.1130

Visualization Parameters

Select Y-axis: Strike Price ($)

Ticker Symbol

Enter Ticker Symbol (e.g., SPY): NVDA

Strike Price Filter Parameters

Minimum Strike Price (% of Spot Price): 80.0

Maximum Strike Price (% of Spot Price): 120.0

Implied Volatility Surface for NVDA Options!

Display with Ticker: NVDA

Discussion

This project was inspired by Mateusz Jastrzębski, who developed a similar Implied Volatility Surface generator. Building the backend for this project was particularly valuable for me, as it allowed me to advance beyond my previous project, a Black-Scholes Model options price calculator. Through the development of the IV Surface Generator, I gained insight into how the Black-Scholes Model can be applied in software development to visualize complex financial data.

Looking ahead, I believe the next steps for this project involve utilizing the IV Surface Generator for arbitrage-based trading strategies. IV surfaces are commonly used to identify patterns in the options market, such as the "U"-shaped curve where implied volatility tends to be higher for both in-the-money (ITM) and out-of-the-money (OTM) options, while being lower for at-the-money (ATM) options. This pattern can suggest increased demand for deep ITM and OTM options due to hedging strategies. Traders could leverage this insight to implement strategies like straddles or strangles, aiming to profit from anticipated future volatility.

My future plans include developing a server capable of capturing snapshots of the implied volatility surface at regular intervals (e.g., every half hour or minute) based on configurable parameters, and storing these in a database. Additionally, I would like to create an interface that allows users to replay these historical IV surfaces, enabling them to observe how the 3D surface evolves over time.

References:

Bland, G. (2020, November 3). *Yfinance Library - A Complete Guide - Algotrading101 blog*. Quantitative Trading Ideas and Guides - AlgoTrading101 Blog. https://algotrading101.com/learn/yfinance-guide/

Butcher, S. (2024, April 18). *Jane Street expanding fast as the average employee makes nearly $3m in profit*. eFinancialCareers. https://www.efinancialcareers.com/news/jane-street-still-expanding-fast-as-the-average-employee-makes-1m-in-profit

Columbia Univeristy. (n.d.). *Interpolation techniques*. Climate Data Library. https://iridl.ldeo.columbia.edu/dochelp/StatTutorial/Interpolation/index.html

Fernando, J. (2024, September 30). *Derivatives: Types, considerations, and pros and cons*. Investopedia. https://www.investopedia.com/terms/d/derivative.asp

Maverick, J. B. (2024, October 10). *How does implied Volatility impact options pricing?*. Investopedia.

https://www.investopedia.com/ask/answers/062415/how-does-implied-volatility-impact-pricing-options.asp

Plotly. (n.d.). *Data apps for production*. Plotly Home. https://plotly.com/

SciPy. (n.d.). *brentq - SciPy v1.14.1 Manual*. SciPy.Optimize.Brentq. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.brentq.html

Stack Overflow. (2021, June 1). *Calculating implied volatility using scipy optimize brentq error*. Stack Overflow. https://stackoverflow.com/questions/66644981/calculating-implied-volatility-using-scipy-optimize-brentq-error

Streamlit. (n.d.). *Streamlit - A faster way to build and share data apps*. Streamlit Framework. https://streamlit.io/

Toulon, Z. (2024, May 20). *JPMorgan accepts less than 1% of intern applications and trains everyone in llms*. JPMorgan accepts less than 1% of intern applications and trains everyone in LLMs. https://www.efinancialcareers.com/news/jpmorgan-acceptance-rate-interns-students