Options Greeks Calculator & Hedging Simulator

A Comprehensive Quantitative Finance Implementation in Python

Eleonora Salcuni
eleonorasalcuni2288@gmail.com

September 29, 2025

Abstract

This project presents an implementation of options pricing and risk management systems using the Black-Scholes model, bringing together knowledge acquired from various courses in stochastic calculus, option pricing, and risk management. The system provides real-time calculation of Greeks, Monte Carlo risk analysis, and portfolio management capabilities. Artificial intelligence was leveraged to create an intuitive and user-friendly interface that allows for the visualization of aggregate portfolio **P&L** and other metrics by adjusting parameters.

Contents

1 Introduction

Options trading and derivative risk management require sophisticated mathematical models and computational frameworks combining Monte Carlo simulation, hedging strategy and evaluating the results with risk performance metrics in order to control the desk/portfolio exposition.

2 Mathematical Framework

2.1 Black-Scholes Model Foundation

The Black-Scholes model serves as the theoretical foundation for option pricing computation. Under the risk-neutral measure, the option value satisfies the fundamental partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{1}$$

where:

$$\begin{aligned}
V(S,t) &= \text{option value as a function of underlying price and time} \\
S &= \text{underlying asset price} \\
t &= \text{time} \\
r &= \text{risk-free interest rate} \\
\sigma &= \text{volatility of the underlying asset}
\end{aligned}$$

The closed-form solutions for European options are:
Call Option Price:

$$C = S_0 N(d_1) - Ke^{-rT} N(d_2) \tag{2}$$

Put Option Price:

$$P = Ke^{-rT} N(-d_2) - S_0 N(-d_1) \tag{3}$$

where:

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \tag{4}$$

$$d_2 = d_1 - \sigma\sqrt{T} \tag{5}$$

and $N(\cdot)$ represents the cumulative standard normal distribution function.

2.2 Options Greeks

The Greeks express the sensitivity of option prices to different parameters, useful both for strategies and for risk measurement.

2.2.1 Delta (Δ) - Price Sensitivity

Delta measures the rate of change of the option price with respect to changes in the underlying asset price:

$$\Delta_{\text{call}} = N(d_1) \tag{6}$$

$$\Delta_{\text{put}} = N(d_1) - 1 = -N(-d_1) \tag{7}$$

Delta represents the hedge ratio and indicates the number of shares of the underlying asset needed to create a risk-neutral hedge.

2.2.2 Gamma (Γ) - Convexity Risk

Gamma measures the rate of change of delta with respect to changes in the underlying price:

$$\Gamma = \frac{\partial \Delta}{\partial S} = \frac{n(d_1)}{S_0 \sigma \sqrt{T}} \tag{8}$$

where $n(\cdot)$ is the standard normal probability density function. Gamma is identical for calls and puts with the same strike and expiration.

2.2.3 Theta (Θ) - Time Decay

Theta measures the rate of change of the option price with respect to time flowing:

$$\Theta_{\text{call}} = -\frac{S_0 n(d_1)\sigma}{2\sqrt{T}} - rKe^{-rT}N(d_2) \tag{9}$$

$$\Theta_{\text{put}} = -\frac{S_0 n(d_1)\sigma}{2\sqrt{T}} + rKe^{-rT}N(-d_2) \tag{10}$$

Theta is typically negative for long positions, representing the erosion of time value.

2.2.4 Vega (ν) - Volatility Sensitivity

Vega measures the sensitivity of the option price to changes in implied volatility:

$$\nu = S_0 n(d_1)\sqrt{T} \tag{11}$$

Vega is identical for calls and puts with the same parameters and is highest for at-the-money options.

2.2.5 Rho (ρ) - Interest Rate Sensitivity

Rho measures the sensitivity of the option price to changes in the risk-free interest rate:

$$\rho_{\text{call}} = KTe^{-rT}N(d_2) \tag{12}$$

$$\rho_{\text{put}} = -KTe^{-rT}N(-d_2) \tag{13}$$

3 System Architecture and Implementation

3.1 Object-Oriented Design

Black and Scholes model is the basis for the option pricing framework and its metrics computation. Based on this, institutional investors and financial institutions use a portfolio of options. Therefore, is truly useful to add all the positions in order to compute the total portfolio greek and pnl. Then, another fundamental feature, is the computation of risk analysis because of the volatility of option market. Metrics implemented in this field are Monte Carlo var, stress testing and gbm simulation. Greeks are also particularly exploited for the hedging or investing strategies and some of the possible strategies are delta-hedge, gamma-scalping and vega-hedge.

3.2 Core Components

3.2.1 BlackScholesModel Class

Implements the complete mathematical framework as static methods, ensuring computational efficiency and mathematical accuracy:

Listing 1: Black-Scholes Implementation Core

```python
@staticmethod
def call_price(S, K, T, r, sigma):
    """Calculate European call option price using Black-Scholes formula."""
    if T <= 0:
        return max(S - K, 0)  # Intrinsic value at expiration

    d1 = BlackScholesModel.d1(S, K, T, r, sigma)
    d2 = BlackScholesModel.d2(S, K, T, r, sigma)

    return S * norm.cdf(d1) - K * np.exp(-r*T) * norm.cdf(d2)
```

3.2.2 Option Class

Encapsulates individual option contracts with dynamic property calculation:

Listing 2: Option Class Design

```python
class Option:
    def __init__(self, S, K, T, r, sigma, option_type='call', quantity=1):
        self.S = S           # Spot price
        self.K = K           # Strike price
        self.T = T           # Time to expiry
        self.r = r           # Risk-free rate
        self.sigma = sigma   # Volatility
        self.option_type = option_type.lower()
        self.quantity = quantity
        self.entry_price = self.price

    @property
    def price(self):
        """Dynamic price calculation."""
        if self.option_type == 'call':
            return BlackScholesModel.call_price(self.S, self.K, self.T, self.r,
                self.sigma)
        else:
```

```python
        return BlackScholesModel.put_price(self.S, self.K, self.T, self.r,
            self.sigma)
```

3.2.3 OptionsPortfolio Class

Manages multi-position portfolios with aggregated risk metrics:

Listing 3: Portfolio Greeks Aggregation

```python
def portfolio_greeks(self):
    """Calculate portfolio-level Greeks through position aggregation."""
    portfolio_delta = sum(pos.delta * pos.quantity for pos in self.positions)
    portfolio_gamma = sum(pos.gamma * pos.quantity for pos in self.positions)
    portfolio_theta = sum(pos.theta * pos.quantity for pos in self.positions)
    portfolio_vega = sum(pos.vega * pos.quantity for pos in self.positions)
    portfolio_rho = sum(pos.rho * pos.quantity for pos in self.positions)

    return {
        'Delta': portfolio_delta,
        'Gamma': portfolio_gamma,
        'Theta': portfolio_theta,
        'Vega': portfolio_vega,
        'Rho': portfolio_rho
    }
```

4 Risk Management Framework

4.1 Value at Risk Implementation

The system implements Monte Carlo Value at Risk (VaR) calculation using geometric Brownian motion for price simulation:

Algorithm 1 Monte Carlo VaR Calculation

Input: Portfolio positions, confidence level α, time horizon T, simulations N
Initialize: Current portfolio value V_0
for $i = 1$ to N **do**
  Generate random price scenario using GBM: $S_T^{(i)} \sim \text{LogNormal}$
  Calculate portfolio value under scenario: $V_T^{(i)}$
  Compute P&L: $\text{PnL}^{(i)} = V_T^{(i)} - V_0$
end for
Sort P&L distribution: $\{\text{PnL}^{(1)}, \text{PnL}^{(2)}, \ldots, \text{PnL}^{(N)}\}$
$\text{VaR}_\alpha = \text{Percentile}(\text{PnL}, (1 - \alpha) \times 100)$
$\text{ES}_\alpha = \mathbb{E}[\text{PnL}|\text{PnL} \leq \text{VaR}_\alpha]$
return VaR, Expected Shortfall

4.2 Stress Testing Framework

The system implements comprehensive stress testing across multiple market scenarios:

Table 1: Stress Testing Scenarios

Scenario	Spot Price Shock	Volatility Shock
Bull Market	+20%	-20%
Bear Market	-20%	+30%
Market Crash	-30%	+50%
Volatility Spike	0%	+50%
Volatility Crush	0%	-50%
Combined Stress	-25%	+40%

5 Hedging Strategy Implementation

5.1 Delta Hedging Algorithm

Delta hedging maintains portfolio delta neutrality through continuous rebalancing:

Algorithm 2 Dynamic Delta Hedging

Initialize: Portfolio with options positions, hedge position $h = 0$
while market is active **do**
  Calculate current portfolio delta: $\Delta_p = \sum_{i=1}^{n} \Delta_i \times q_i$
  Determine required hedge adjustment: $\Delta h = -\Delta_p - h$
  Execute hedge trade in underlying asset: $h \leftarrow h + \Delta h$
  Record hedging transaction and portfolio metrics
  Update market data and recalculate Greeks
  Wait for next rebalancing interval
end while

5.2 Gamma Scalping Strategy

Gamma scalping exploits the difference between realized and implied volatility:

$$\text{Gamma P\&L} = \frac{1}{2} \times \Gamma_{portfolio} \times (\Delta S)^2 \tag{14}$$

The strategy profits when realized volatility exceeds implied volatility embedded in option prices.

6 User Interface Design

6.1 Command-Line Interface

The CLI provides comprehensive analysis output including:

Listing 4: Sample CLI Output Structure

```
============================================================
OPTIONS GREEKS CALCULATOR & HEDGING SIMULATOR
============================================================

1. PORTFOLIO SUMMARY
    Position details with individual Greeks calculation
```

```

2. PORTFOLIO GREEKS
    Aggregated risk metrics across all positions

3. RISK ANALYSIS
    Monte Carlo VaR and Expected Shortfall

4. STRESS TESTING
    Portfolio performance under adverse scenarios

5. HEDGING SIMULATION
    Dynamic hedging strategy backtesting

6. MONTE CARLO ANALYSIS
    Price simulation and statistical analysis
============================================================
```

6.2 Web Application Interface

The Streamlit-based web application provides real-time interactive analysis:

- **Parameter Control**: Sidebar sliders for real-time adjustment
- **Sensitivity Analysis**: Dynamic Greeks visualization
- **Payoff Diagrams**: Interactive P&L charts
- **Monte Carlo Module**: Configurable risk simulation
- **Portfolio Manager**: Multi-position analysis interface

7 Performance Analysis and Validation

7.1 Computational Complexity

Table 2: Algorithmic Complexity Analysis

Operation	Time Complexity
Single Option Pricing	$O(1)$
Portfolio Greeks Calculation	$O(n)$
Monte Carlo VaR (N simulations)	$O(N \times n)$
Stress Testing (S scenarios)	$O(S \times n)$
Dynamic Hedging (T time steps)	$O(T \times n)$

where n represents the number of positions in the portfolio.

7.2 Numerical Accuracy

The implementation incorporates several numerical stability enhancements:

- Edge case handling for zero time to expiry

- Numerical integration for complex payoff structures
- Adaptive time stepping in simulation algorithms
- Error bounds verification for Monte Carlo convergence

8 Results and Case Study Analysis

8.1 Sample Portfolio Configuration

Consider a representative portfolio with the following positions:

- Long 10 Call options: $S = 100$, $K = 105$, $T = 0.25$, $\sigma = 20\%$
- Long 5 Put options: $S = 100$, $K = 95$, $T = 0.25$, $\sigma = 20\%$

8.2 Risk Metrics Results

Table 3: Portfolio Risk Analysis Results

Risk Metric	**Value**
Portfolio Delta	2.5430
Portfolio Gamma	0.5373
Portfolio Theta	-0.3248
Portfolio Vega	2.6866
Portfolio Rho	0.5546
95% VaR (1-day)	-$3.87
Expected Shortfall	-$4.48
Maximum Drawdown	-$12.67

8.3 Stress Testing Results

Table 4: Portfolio Performance Under Stress Scenarios

Scenario	**Spot Change**	**Vol Change**	**P&L (%)**
Bull Market	+20%	-20%	+406%
Bear Market	-20%	+30%	+125%
Market Crash	-30%	+50%	+269%
Vol Spike	0%	+50%	+86%
Vol Crush	0%	-50%	-75%
Combined Stress	-25%	+40%	+196%

The results demonstrate the portfolio's positive convexity characteristics, with profits in most stress scenarios except volatility crush.

9 Software Engineering Best Practices

9.1 Testing and Validation

Quality assurance measures include:

- Unit tests for Black-Scholes pricing accuracy
- Greeks calculation verification using finite differences
- Portfolio aggregation correctness validation
- Monte Carlo convergence testing
- Edge case handling verification

10 Conclusions and Future Extensions

10.1 Project Achievements

This implementation successfully demonstrates:

- Complete quantitative finance framework implementation
- Advanced risk management and portfolio analysis capabilities
- Professional-grade visualization and user interface design
- Scalable architecture suitable for extension and enhancement
- Integration of theoretical concepts with practical trading applications

10.2 Future Enhancement Opportunities

Potential extensions include:

1. **Advanced Models**: Implementation of stochastic volatility models (Heston, SABR)
2. **Exotic Options**: Support for barrier, Asian, and lookback options
3. **Multi-Asset Options**: Basket options and correlation trading
4. **Machine Learning Integration**: Volatility forecasting and pattern recognition
5. **Real-Time Data Integration**: Market data feeds and live trading capabilities
6. **Performance Optimization**: GPU acceleration for Monte Carlo simulations
7. **Database Integration**: Historical data storage and backtesting frameworks

11 References

1. Hull, John C. *Options, Futures, and Other Derivatives*. 10th Edition, Pearson, 2018.
2. Black, Fischer, and Myron Scholes. "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy* 81, no. 3 (1973): 637-654.
3. Wilmott, Paul. *Paul Wilmott Introduces Quantitative Finance*. 2nd Edition, Wiley, 2007.
4. Glasserman, Paul. *Monte Carlo Methods in Financial Engineering*. Springer, 2004.
5. Taleb, Nassim Nicholas. *Dynamic Hedging: Managing Vanilla and Exotic Options*. Wiley, 1997.
6. Rebonato, Riccardo. *Volatility and Correlation: The Perfect Hedger and the Fox*. 2nd Edition, Wiley, 2004.

A Code Repository Structure

Listing 5: Complete Project Structure

```
options-greeks-project/
          main.py                      # Command-line interface
          app.py                       # Streamlit web application
          requirements.txt             # Python dependencies
          README.md                    # Project documentation
          docs/                        # Documentation directory
                project_report.pdf       # This LaTeX document
                api_reference.md         # API documentation
                user_guide.md           # Usage instructions
          tests/                       # Unit tests directory
                test_black_scholes.py    # Pricing model tests
                test_portfolio.py        # Portfolio management tests
                test_risk_analysis.py    # Risk calculation tests
          examples/                    # Usage examples
                basic_usage.py           # Simple examples
                portfolio_analysis.py    # Advanced portfolio analysis
                hedging_strategies.py    # Hedging implementation examples
          output/                      # Generated charts and reports
              payoff_diagram.png        # Portfolio payoff charts
              greeks_sensitivity.png    # Greeks analysis charts
              monte_carlo_simulation.png # Risk simulation results
              hedging_performance.png   # Hedging strategy results
```