NeuroHedg: A Practical Baseline for Option Pricing and Hedging Beyond Black Scholes (Phase 1 Report)

NeuroHedg Project Team

October 14, 2025

Contents

1 Executive Summary

Financial derivatives like options are the plumbing of modern markets, but pricing them correctly is a famously leaky affair. The Black-Scholes (BS) model has been a cornerstone in quantitative finance for decades. Yet, it rests on assumptions that don't hold in the wild: volatility is constant, markets are frictionless, and hedging is continuous. Phase 1 of NeuroHedg tackles this head-on by building a clean, reliable, and testable implementation from scratch—not just to replicate the textbook, but to rigorously map its limits.

This document describes the full journey: deriving call/put prices and Greeks, solving for implied volatility, and validating against reference implementations. It captures both the mathematical foundations and the numerical traps, packaging the design choices that shaped the code. The ultimate goal is a solid baseline written in Python, understandable by curious minds alike. Future phases will go beyond the Black-Scholes model.

2 Project Scope & Contributions

This initial phase focused on delivering the following core components:

- **Analytical Pricers:** Clean implementations for European puts and calls.
- **The Greeks:** Sensitivity calculations for Delta, Gamma, Vega, Theta, and Rho.
- **Implied Volatility Solver:** A robust, bisection-style solver to back out volatility from a market price.
- **Unit Tests:** Comprehensive tests to ensure correctness, including IV recovery and no-arbitrage checks like put-call parity.
- **Visualizations:** Plots to generate intuition around key option pricing concepts.

Out of Scope for Phase 1: Stochastic volatility models, integration with real market data, or a front-end application UI.

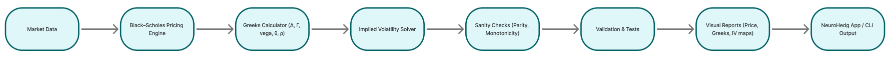
3 System Overview

3.1 Architecture

The project is structured logically to separate concerns. The data flows from user inputs through the pricer to the derived sensitivities in an integrated and testable manner.

```
project-root/
          src/
                bs/           # Core Black-Scholes pricing logic
                utils/        # Helper functions and numerical utilities
                invariants/   # Financial and mathematical checks
          tests/            # Unit tests
```

3.2 Flow Diagram Illustration


Market Data
Black–Scholes Pricing Engine
Greeks Calculator (Δ, Γ, vega, θ, ρ)
Implied Volatility Solver
Sanity Checks (Parity, Monotonicity)
Validation & Tests
Visual Reports (Price, Greeks, IV maps)
NeuroHedg App / CLI Output

4 The Black-Scholes Model: Formulas

The model provides a closed-form solution for European options, assuming the underlying asset follows a Geometric Brownian Motion. The price is a function of: Spot Price (S), Strike Price (K), Time to Maturity (T), Risk-Free Rate (r), and Volatility (σ).

The price of a European call option C and a European put option P are given by:

$$C(S,T) = N(d_1)S - N(d_2)Ke^{-rT}$$

$$P(S,T) = N(-d_2)Ke^{-rT} - N(-d_1)S$$

where $N(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution, and d_1 and d_2 are:

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

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

5 Sanity Checks & Validation

Rigorous validation was performed to ensure the model's correctness.

5.1 Reference Values

The implementation was validated against known reference values.

Table 1: Reference Greeks for a European Option

Parameter	**Value**
Inputs: $S = 100$, $K = 100$, $r = 5\%$, $\sigma = 20\%$, $T = 1$ *year*	
Call Price	10.4506
Put Price	5.5735
Delta (Call)	0.6368
Gamma	0.0199
Vega	39.669
Theta (Call, yr)	-6.41
Rho (Call)	53.232

5.2 Core Invariants

The implementation respects fundamental no-arbitrage relationships like Put-Call Parity.

$$C - P = S - Ke^{-rT}$$

Table 2: Invariant Checks

Check	**Description**	**Result**
Put-Call Parity	The relationship $C - P = S - Ke^{-rT}$ must hold.	Passed
Monotonicity	Call prices decrease as strike K increases; puts are opposite.	Passed
Greeks Signs	Key greeks must have the correct sign (e.g., Vega > 0).	Passed
IV Solver	Must accurately recover input volatility from the calculated price.	Passed

6 The Engineering Journey

The path from concept to code involved several practical software engineering challenges:

- **Package Management:** Adopting the modern `src/` layout for clean project setup.
- **Development Workflow:** Establishing a reproducible environment using editable installs.

  ```
  pip install -e .[dev]
  ```

- **Numerical Paranoia:** Implementing guards against floating-point inaccuracies, especially for edge cases like near-zero time to expiry.

Each obstacle was treated as a learning opportunity that informed the final design.

7 Real-World Impact

This foundational work is intended to be a resource for:

- **Education:** Teaching option pricing with a transparent implementation.
- **Benchmarking:** Serving as a reliable baseline for comparing more advanced models.
- **Research & Development:** Providing a modifiable foundation for exploring new models.