OptionMC: A Python Package for Monte Carlo Pricing of European Options

Sandy H. S. Herho [a,1,*], Siti N. Kaban [b], Cahya Nugraha [c]

[a] Department of Earth and Planetary Sciences, University of California, Riverside, California, United States
[b] Financial Engineering Program WorldQuant University Washington, D.C., United States
[c] Al Mumtaaz Islamic Charitable Foundation, Karawang, West Java, Indonesia
[1] sandy.herho@email.ucr.edu
* corresponding author

ARTICLE INFO

Article history
Received April 26, 2025
Revised October 7, 2025
Accepted December 8, 2025

Keywords
Antithetic variates
Financial derivatives
Monte Carlo methods
Option pricing
Stochastic simulation

ABSTRACT

This article presents OptionMC, a Python package designed for educational purposes that implements Monte Carlo methods for European option pricing. We describe the package's architecture and demonstrate its application through systematic testing against established Black-Scholes analytical solutions. The implementation supports both standard Monte Carlo estimation and variance reduction via antithetic variates, allowing examination of convergence patterns and computational efficiency. Our results suggest that Monte Carlo estimates converge toward analytical solutions as the number of iterations increases, with convergence behavior generally consistent with theoretical expectations. Analysis of parameter sensitivity indicates the package appropriately captures fundamental pricing relationships, including volatility effects, time decay, and moneyness considerations. The distributional characteristics of simulated stock prices and option payoffs align reasonably well with theoretical predictions. While OptionMC primarily serves pedagogical objectives rather than high-performance applications, it offers a transparent framework that may benefit students and researchers seeking to understand the practical implementation of option pricing algorithms through Monte Carlo techniques.

1. Introduction

Financial derivatives have become increasingly important in modern financial markets, with options contracts constituting one of their most essential and widely traded forms. These sophisticated instruments enable participants to hedge against unfavorable price movements, speculate on directional market shifts, and construct complex trading strategies with asymmetric risk-return profiles [1]. The accurate valuation of such contracts is central to virtually all financial operations, from risk management and regulatory reporting to portfolio optimization and algorithmic trading [2]. In this context, computational methods for option pricing have undergone significant evolution over the past several decades, with Monte Carlo simulation establishing itself as one of the most flexible and widely applicable approaches for handling complex derivative instruments [3], [4].

The foundations of modern option pricing theory were established in 1973 with the publication of two landmark papers. Black and Scholes [5] proposed their eponymous partial differential equation model for European option valuation, while Merton [6] extended this framework to incorporate dividend payments and developed the theory of rational option pricing. Their collective work

revolutionized financial markets by establishing a rigorous mathematical approach to derivative pricing based on no-arbitrage principles, thereby transforming options from speculative vehicles into precisely valued financial instruments. This breakthrough earned Scholes and Merton the 1997 Nobel Prize in Economics, with Black being acknowledged posthumously.

Despite its theoretical elegance and computational efficiency, the Black-Scholes framework suffers from well-documented shortcomings when applied to real-world market conditions. The model's assumptions of constant volatility, log-normal distribution of returns, continuous trading, perfect liquidity, and zero transaction costs rarely hold in practice [4], [7]. These limitations manifest most acutely during periods of market stress, when implied volatilities often exhibit pronounced skewness (smiles) and return distributions exhibit fatter tails than those predicted by a normal distribution. More complex features, such as path dependence, multiple underlying assets, or early exercise rights, further complicate analytical solutions, necessitating more flexible numerical approaches.

Among these numerical methods, Monte Carlo simulation has emerged as particularly valuable, offering unparalleled versatility in handling complex derivative structures. First applied to option pricing by Boyle et al. [3], this technique enables the valuation of instruments where closed-form solutions are unavailable or unwieldy. The fundamental approach involves generating numerous random price trajectories for the underlying asset under a specified stochastic process, computing the option payoff for each path, and then averaging the discounted payoffs to estimate the option's value. The methodology's greatest strength lies in its ability to accommodate virtually any payoff structure or price process, including jumps, stochastic volatility, or mean reversion, making it indispensable for pricing exotic and path-dependent options [4], [8].

The conceptual simplicity of Monte Carlo methods, however, belies significant practical challenges in their implementation. Principal among these is computational efficiency, as the standard error of Monte Carlo estimators decreases at a rate proportional to the square root of the number of simulations. Consequently, achieving a tenfold improvement in precision requires a hundredfold increase in the simulation budget, imposing substantial computational demands for high-accuracy valuations [9]. This inherent limitation has spurred extensive research into variance reduction techniques, which aim to enhance estimation efficiency without proportionally increasing computational cost. These methods exploit analytical information or statistical properties of the simulation to reduce variability in the estimator, thereby achieving greater precision with fewer simulations [4], [7].

Among these variance-reduction approaches, antithetic variates warrant particular attention for their exceptional combination of simplicity and effectiveness. This technique involves generating negatively correlated pairs of sample paths by using the negative of each random shock alongside the original drawing. When the payoff function is monotonic or nearly so with respect to these shocks, the resulting negative correlation between path pairs reduces the estimator's overall variance. Despite its straightforward implementation, antithetic sampling often yields substantial efficiency gains, reducing the number of simulations required by factors of 2 to 5 without significantly increasing per-path computational complexity [4], [10]. While more sophisticated techniques, such as control variates or importance sampling, can achieve greater variance reduction in specific contexts, antithetic variates remain widely used due to their robust performance across diverse option types and minimal additional implementation complexity [7].

Recent years have witnessed substantial advances in the computational infrastructure supporting Monte Carlo methods in finance. Xiong et al. [11] has demonstrated significant efficiency improvements through distributed computing architectures applied to Least Squares Monte Carlo methods for American option pricing. These approaches decompose the high-dimensional simulation problem into more manageable subproblems that can be solved in parallel, effectively leveraging modern multi-processor systems. Even more dramatic innovations are emerging at the intersection of quantum computing and computational finance, where Rebentrost et al. [12] have proposed quantum algorithms for Monte Carlo pricing that theoretically offer quadratic speedups relative to

classical methods. While practical implementation of quantum advantage remains on the horizon, these developments highlight the continuing evolution of computational methods for option pricing.

The pedagogical dimension of option pricing methodology warrants particular consideration. Financial engineering and computational finance curricula increasingly emphasize hands-on implementation of pricing models alongside theoretical foundations, reflecting the practical orientation of these disciplines [2]. Interactive computational tools have demonstrated significant efficacy in enhancing students' comprehension of complex financial concepts, enabling experimentation with parameter values and direct observation of their effects on pricing outputs [13]. Among programming environments, Python has emerged as an especially valuable platform for financial education due to its readable syntax, extensive mathematical libraries, and visualization capabilities, which lower barriers to learning complex concepts [14], [15]. Despite being a high-level language, Python's scientific computing stack, powered by optimized libraries such as NumPy, SciPy, and Numba, delivers competitive performance for numerical tasks, often rivalling languages such as C++ or Java when leveraging vectorized operations or parallel processing [16], [17], [18], [19]. Additionally, Python's rich ecosystem (e.g., pandas for data manipulation, Matplotlib/Plotly for visualization, and Jupyter for interactive workflows) and seamless integration with tools such as TensorFlow and QuantLib make it a versatile, end-to-end solution for both educational and real-world financial modeling [20], [21].

This pedagogical imperative aligns with a broader movement toward open-source financial software development within both academic and industry contexts. Such initiatives democratize access to sophisticated pricing methodologies, enabling broader adoption across educational institutions and smaller financial firms that might otherwise lack resources for proprietary trading systems [22]. Beyond mere accessibility, open-source implementations foster reproducible research and facilitate rigorous validation of pricing approaches across diverse market conditions. The transparency inherent in open-source development allows practitioners to inspect algorithms directly, enhancing trust in computational outcomes and enabling customization for specific requirements [13].

The proliferation of option pricing libraries has not, however, uniformly addressed the need for implementations that balance educational clarity with computational rigor. Many existing packages prioritize performance at the expense of intelligibility, implementing optimization techniques that obscure the fundamental algorithms. Others focus on breadth of functionality rather than depth of exposition, providing cursory implementations of numerous pricing methods without illuminating their mathematical foundations or computational nuances. The academic literature increasingly recognizes this tension between efficiency and transparency, with several authors advocating for implementations that serve both pedagogical and practical purposes [13], [23].

In response to these considerations, we introduce OptionMC, a Python package for European option pricing via Monte Carlo simulation. The package implements both standard Monte Carlo estimation and variance-reduced simulation using antithetic variates, with all implementations accompanied by comprehensive documentation explaining the mathematical foundations and computational design choices. Beyond merely calculating point estimates, OptionMC provides extensive visualization tools for exploring convergence behavior, parameter sensitivity, and the distribution of option payoffs. Comparisons with Black-Scholes analytical solutions are integrated throughout, enabling direct assessment of numerical accuracy for standard options while illuminating the flexibility of simulation methods for more complex structures.

Unlike packages that emphasize raw performance or algorithmic sophistication, OptionMC deliberately prioritizes educational value and implementation clarity, making it particularly suitable for teaching and research environments. The code structure reflects the mathematical structure of the underlying models, with a clear separation between simulating the price process and calculating payoffs and applying variance-reduction techniques. This modular design facilitates both conceptual understanding and practical experimentation, allowing users to modify individual components while preserving overall functionality. Through this approach, OptionMC addresses the gap between

theoretical exposition and practical implementation that often challenges students and researchers in quantitative finance.

2. Methods

2.1 Black-Scholes Model for European Options

The Black-Scholes option pricing model represents a cornerstone of modern financial theory, originating from the seminal work of Black and Scholes [5] and Merton [6]. The model rests upon a specific set of idealized market conditions: frictionless markets without transaction costs, continuous trading opportunities, constant risk-free interest rate and volatility, and log-normally distributed asset prices. These assumptions, while simplifying, permit the derivation of a closed-form solution for European option valuation that remains fundamental to quantitative finance despite well-documented limitations [24], [25].

The mathematical foundation begins with a specification of the underlying asset price dynamics. Under the Black-Scholes framework, the price of the underlying asset S_t follows a geometric Brownian motion, expressed through the stochastic differential equation:

$$dS_t = \mu S_t dt + \sigma S_t dW_t. \tag{1}$$

In this formulation, μ represents the expected return or drift of the asset, σ denotes the volatility or standard deviation of returns, and W_t is a standard Wiener process (Brownian motion). This particular structure captures the empirically observed behavior that asset returns tend to be generally distributed while ensuring that prices remain strictly positive [1]. The term $\mu S_t dt$ represents the deterministic component of price changes, reflecting the expected growth proportional to the current price, while $\sigma S_t dW_t$ models the random fluctuations whose magnitude scales with the asset price.

To derive the option pricing equation, we consider a derivative security $V(S_t, t)$ whose value depends on the underlying asset price and time. According to Ito's lemma, a fundamental result in stochastic calculus, the differential of this derivative's value can be expressed as:

$$dV = \left(\frac{\partial V}{\partial t} + \mu S_t \frac{\partial V}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S_t^2}\right) dt + \sigma S_t \frac{\partial V}{\partial S_t} dW_t. \tag{2}$$

The critical insight of Black, Scholes, and Merton was to construct a risk-free portfolio by combining the derivative and the underlying asset in specific proportions. This portfolio, denoted Π, consists of shorting one unit of the derivative and holding $\frac{\partial V}{\partial S_t}$ units of the underlying asset, yielding a value:

$$\Pi = -V + \frac{\partial V}{\partial S_t} S_t. \tag{3}$$

The change in this portfolio's value over an infinitesimal time period is determined by computing the differential:

$$d\Pi = -dV + \frac{\partial V}{\partial S_t} dS_t. \tag{4}$$

Substituting equations (1) and (2) into (4) and collecting terms yields:

$$d\Pi = -\left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S_t^2}\right) dt. \tag{5}$$

The remarkable feature of equation (5) is the absence of the stochastic term dW_t, meaning this portfolio is instantaneously riskless. The fundamental principle of no-arbitrage requires that such a riskless portfolio must earn precisely the risk-free rate of return, expressed mathematically as:

$$d\Pi = r\Pi dt, \tag{6}$$

where r represents the constant risk-free interest rate. Combining equations (3), (5), and (6), and rearranging terms leads to the celebrated Black-Scholes partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S_t^2} + rS_t \frac{\partial V}{\partial S_t} - rV = 0. \tag{7}$$

This second-order parabolic partial differential equation determines the evolution of any derivative security's price in the Black-Scholes framework. For European options, the PDE is solved subject to the appropriate terminal boundary condition. In the case of a European call option, this condition is $V(S_T, T) = max(S_T - K, 0)$, while for a European put option, it is $V(S_T, T) = max(K - S_T, 0)$, where K represents the strike price and T is the expiration time [4].

The solution to equation (7) with the specified boundary conditions yields the famous Black-Scholes formulas for European option prices. For a European call option, the price at time t is given by:

$$C(S_t, t) = S_t \Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2). \tag{8}$$

Similarly, for a European put option, the price is:

$$P(S_t, t) = Ke^{-r(T-t)}\Phi(-d_2) - S_t\Phi(-d_1). \tag{9}$$

In these formulas $\Phi(.)$, represents the cumulative distribution function of the standard normal distribution, and the parameters d_1 and d_2 are defined as:

$$d_1 = \frac{ln\, ln\, (S_t/K) \; + (r + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}}, \tag{10}$$

$$d_2 = d_1 - \sigma\sqrt{T - t}. \tag{11}$$

The equations (8) and (9) provide a direct and computationally efficient means of pricing European options. This analytical tractability represents one of the model's most significant advantages and explains its enduring popularity despite known limitations. The formula can be interpreted probabilistically: under the risk-neutral measure, $\Phi(d_2)$ represents the probability that the option expires in the money, while $S_t\Phi(d_1)/Ke^{-r(T-t)}\Phi(d_2)$ gives the conditional expectation of the asset price at expiration, given that the option expires in-the-money [1].

Despite its mathematical elegance, the Black-Scholes model exhibits several limitations that stem from its assumptions. The constant-volatility assumption contradicts empirical observations of "volatility smiles" and "term structures" in options markets, in which implied volatilities vary with strike price and time to expiration [26,27]. This phenomenon became particularly pronounced after the 1987 market crash, which revealed that traders assign higher probabilities to extreme market movements than the log-normal distribution would suggest [28]. Additionally, the model's assumption of continuous asset price paths fails to capture sudden jumps observed in practice, leading to systematic mispricing of deep out-of-the-money options [29].

These limitations have motivated the development of more sophisticated models, including stochastic volatility models like Heston's [30], implied volatility trees [31,32], and local volatility models [33,34]. Recent empirical research has explored how market dynamics differ across various quantiles of the price distribution, revealing that price movements in the upper and lower tails exhibit distinct patterns not captured by the Black-Scholes framework [24]. Moreover, machine learning approaches have gained traction as alternatives to traditional parametric models, owing to their flexibility in capturing complex patterns in market data without relying on restrictive assumptions [35].

In the OptionMC package, we implement the Black-Scholes analytical solution in the bs_analytical_price method of the OptionPricing class. This implementation uses the cumulative normal distribution function provided by the SciPy library [36] to calculate $\Phi(d_1)$ and $\Phi(d_2)$

efficiently. The analytical solution serves as both a benchmark for evaluating the accuracy of Monte Carlo simulation results and a demonstration of the foundational theory of option pricing, making it particularly valuable for educational purposes

2.2 Monte Carlo Implementation for Option Pricing

The Monte Carlo method for option pricing involves simulating numerous random price paths for the underlying asset, computing the option payoff for each path, and then averaging these payoffs to estimate the option price. This section details our implementation of both standard Monte Carlo simulation and the variance reduction technique using antithetic variates in the OptionMC package.

2.2.1 Standard Monte Carlo Simulation

The foundation of Monte Carlo option pricing lies in the risk-neutral valuation principle, which states that the price of an option equals the expected discounted payoff under the risk-neutral measure. For European options, this can be expressed mathematically as:

$$V_0 = e^{-rT} E^Q[h(S_T)], \tag{12}$$

where V_0 is the option price at time $0, r$ is the risk-free interest rate, T is the time to maturity, E^Q denotes the expectation under the risk-neutral measure Q, and $h(S_T)$ represents the option payoff function at maturity [4].

Under the risk-neutral measure, the underlying asset price follows a geometric Brownian motion with drift equal to the risk-free rate:

$$dS_t = rS_t dt + \sigma S_t dW_t^Q. \tag{13}$$

where W_t^Q is a standard Brownian motion under the risk-neutral measure Q. This stochastic differential equation has an analytical solution given by:

$$S_T = S_0 exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sqrt{T}Z\right), \tag{14}$$

where $Z \sim N(0,1)$ is a standard normal random variable [1].

For a European call option with strike price K, the payoff function is:

$$h(S_T) = max\ (S_T - K, 0). \tag{15}$$

Similarly, for a European put option, the payoff function is:

$$h(S_T) = max\ (K - S_T, 0). \tag{16}$$

The Monte Carlo method approximates the expectation in equation (12) by generating n independent samples of the terminal stock price $S_T^{(i)}$ using equation (14), calculating the corresponding payoffs $h\left(S_T^{(i)}\right)$, and then taking their average [3]:

$$\hat{V}_0 = e^{-rT} \frac{1}{n} \sum_{i=1}^{n} \quad h\left(S_T^{(i)}\right). \tag{17}$$

Our implementation in the OptionMC package follows this approach. For example, the call_option_simulation method in the OptionPricing class generates terminal stock prices using equation (14) and then applies the call option payoff function (15):

$$\hat{C}_0 = e^{-rT} \frac{1}{n} \sum_{i=1}^{n} \quad max\left(S_T^{(i)} - K, 0\right), \tag{18}$$

where $S_T^{(i)} = S_0 exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}Z^{(i)}\right)$ with $Z^{(i)} \sim N(0,1)$.

Similarly, the put_option_simulation method implements the estimator for put options:

$$\hat{P}_0 = e^{-rT}\frac{1}{n}\sum_{i=1}^{n} \quad max\left(K - S_T^{(i)}, 0\right). \tag{19}$$

The standard error of the Monte Carlo estimator is given by [9]:

$$SE\left(\hat{V}_0\right) = e^{-rT}\frac{\sigma_h}{\sqrt{n}}, \tag{20}$$

where σ_h is the standard deviation of the payoff function. This error decreases at a rate proportional to $\sqrt{n}$, which means that to reduce the error by a factor of 10, we need to increase the number of simulations by a factor of 100, making Monte Carlo methods computationally intensive for high-precision applications [7].

2.2.2 Antithetic Variates for Variance Reduction

To improve the efficiency of Monte Carlo simulations, we implement the antithetically varied technique, a variance reduction method that exploits negative correlation between pairs of sample paths. The key insight is that if the random variable Z has a symmetric distribution around zero (like the standard normal distribution), then $-Z$ has the same distribution as Z. For each random draw $Z^{(i)}$, we generate an "antithetic" counterpart $-Z^{(i)}$, resulting in pairs of negatively correlated stock price paths [4].

The terminal stock prices for these antithetic pairs are:

$$\begin{aligned} S_T^{(i)} &= S_0 exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}Z^{(i)}\right), \\ \tilde{S}_T^{(i)} &= S_0 exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}\left(-Z^{(i)}\right)\right). \end{aligned} \tag{21}$$

The antithetic variates estimator for a call option is then [8]:

$$\hat{C}_0^A = e^{-rT}\frac{1}{2n}\sum_{i=1}^{n} \quad \left[max\left(S_T^{(i)} - K, 0\right) + max\left(\tilde{S}_T^{(i)} - K, 0\right)\right]. \tag{22}$$

Similarly, for a put option:

$$\hat{P}_0^A = e^{-rT}\frac{1}{2n}\sum_{i=1}^{n} \quad \left[max\left(K - S_T^{(i)}, 0\right) + max\left(K - \tilde{S}_T^{(i)}, 0\right)\right]. \tag{23}$$

The variance of the antithetic estimator is [9]:

$$Var\left(\hat{V}_0^A\right) = \frac{1}{4}e^{-2rT}\frac{Var\left(h(S_T)\right) + Var\left(h\left(\tilde{S}_T\right)\right) + 2Cov\left(h(S_T), h\left(\tilde{S}_T\right)\right)}{n}. \tag{24}$$

Since the payoff functions for both call and put options are monotonic with respect to the terminal stock price, and the terminal stock prices S_T and $\tilde{S}_T$ are negatively correlated, the covariance term $Cov\left(h(S_T), h\left(\tilde{S}_T\right)\right)$ is negative. This makes the variance of the antithetic estimator smaller than that of the standard Monte Carlo estimator, resulting in more accurate price estimates for the same number of random draws [10].

In our implementation, the antithetic_call_simulation and antithetic_put_simulation methods in the OptionPricing class generate half the number of random variables compared to the standard methods but use both the original and negatively signed versions to create a full set of simulations. This approach is computationally efficient as it requires generating only $n/2$ random numbers for n price paths, while potentially achieving a significant reduction in variance [4].

The efficiency gain from antithetic variates can be quantified using the variance reduction ratio:

$$VRR = \frac{Var(\hat{V}_0)}{Var(\hat{V}_0^A)}. \quad (25)$$

Empirical results with our implementation show that antithetic variates typically achieve a VRR between 2 and 5 for European options, meaning that to achieve the same level of accuracy, the antithetic method requires 2 to 5 times fewer simulations than the standard method [10].

Furthermore, the confidence intervals for the Monte Carlo estimates can be calculated using the central limit theorem [4]:

$$\hat{V}_0 \pm z_{\alpha/2}.SE(\hat{V}_0), \quad (26)$$

where $z_{\alpha/2}$ is the critical value from the standard normal distribution for a given confidence level $1-\alpha$. In our package, the confidence_intervals method implements this calculation, providing users with a measure of the precision of the Monte Carlo estimates.

The implementation in OptionMC ensures that all these mathematical principles are applied correctly while maintaining code clarity for educational purposes. By separating the price process simulation from the payoff calculation and variance reduction techniques, the package allows users to understand each component individually and observe how they interact to produce option price estimates [23].

2.3 Package Demonstration and Applications

The OptionMC package provides tools for option pricing using Monte Carlo simulation. This section describes the methodological approaches for applying the package in financial analysis and education.

2.3.1 Basic Option Pricing

The core methodology of OptionMC centers on pricing European options through Monte Carlo simulation with comparative analysis against analytical Black-Scholes solutions. The implementation workflow consists of defining option parameters, initializing the pricing model, and executing the simulation. The OptionPricing class accepts parameters including the initial stock price (S_0), strike price (K), time to maturity (T), risk-free rate (r), volatility (σ), and the number of simulation iterations.

Our simulation generates random stock price paths under geometric Brownian motion, computes option payoffs at maturity, and discounts the average to estimate the option price. The package provides functionality to retrieve both the simulated stock price paths and corresponding option payoffs for further analysis.

For visualization, the package provides methods to generate distributions of terminal stock prices and option payoffs. The OptionVisualizer class handles the creation of these visualizations with configurable parameters for titles, labels, and output formats.

2.3.2 Variance Reduction Techniques

The package implements variance reduction through the antithetic variates method. The methodology involves comparing a standard Monte Carlo simulation with the antithetic approach through parallel implementations in the call_option_simulation and antithetic_call_simulation

methods. To analyze convergence behavior, the package enables systematic testing across varying iteration counts. The methodology involves tracking option prices, computation times, and relative errors as the number of iterations increases. The confidence_intervals method provides statistical bounds on the Monte Carlo estimates based on the central limit theorem. The computational efficiency comparison methodology measures both the accuracy relative to analytical solutions and the execution time for each method. This approach enables quantification of the efficiency gains achieved by variance reduction techniques.

2.3.3 Parameter Sensitivity Analysis

OptionMC provides methodological tools for parameter sensitivity analysis by systematically varying key inputs. The methodology involves defining a range of values for a specific parameter (such as volatility, time to maturity, or interest rate), computing option prices for each value while holding other parameters constant, and analyzing the resulting price patterns. The volatility sensitivity analysis methodology systematically varies the volatility parameter within a specified range while holding other inputs constant. The same approach applies to time-to-maturity and interest rate sensitivity analyses, enabling examination of their respective effects on option valuation. The package's plot_parameter_sensitivity method generates visualizations of these parameter relationships, with options to overlay analytical solutions for comparison. The methodology includes storing all simulation parameters and results in data files, ensuring reproducibility and facilitating external analysis.

2.3.4 Moneyness and Accuracy Analysis

The package includes a methodology for analyzing how the accuracy of Monte Carlo simulations varies with option moneyness and iteration count. This approach involves systematically varying the strike price across a range of values to create options with different moneyness levels (the ratio of strike price to current stock price). The methodological process involves defining a range of strike prices and a set of iteration counts, then computing option prices using both Monte Carlo simulation and analytical solutions for each parameter combination. The relative pricing error is calculated for each case, allowing for analysis of how accuracy varies across the moneyness spectrum.

To quantify the relationship between accuracy and simulation size, the methodology incorporates multiple iteration counts for each moneyness level. This approach enables assessment of the convergence rate as simulation size increases and identification of potential differences in convergence behavior across moneyness levels. The package provides functions to categorize options by moneyness (in-the-money, at-the-money, out-of-the-money) and compute summary statistics for each category. This methodology facilitates structured analysis of the relationship between option characteristics and simulation accuracy. These methodological approaches demonstrate how OptionMC can be applied for both educational and analytical purposes in financial engineering. The package provides systematic frameworks for exploring various aspects of option pricing theory and the practical considerations of implementing Monte Carlo methods

3. Results and Discussion

The Monte Carlo convergence analysis for European option pricing demonstrates that price estimates stabilize as the simulation size increases. As shown in Figure 1, with a small number of iterations (10^2), the Monte Carlo estimate exhibits substantial variability, with a price estimate of approximately \$8.15 and wide 95% confidence intervals spanning from roughly \$5.75 to \$10.55. This initial imprecision is a direct consequence of the stochastic nature of the simulation process and the inherent sampling error with limited data points. The convergence pattern follows the expected behavior dictated by the central limit theorem, with the estimate approaching the analytical value at a rate proportional to $\sqrt{n}$, where n is the number of iterations [9]. At approximately 10^3 iterations, the estimate has already improved significantly to around \$10.15, with a narrower confidence interval. Beyond 10^4 iterations, the Monte Carlo price converges to the analytical Black-Scholes price of \$10.45, with the confidence interval narrowing to less than \$0.5. At 10^5 iterations, the Monte Carlo price demonstrates excellent agreement with the analytical solution.

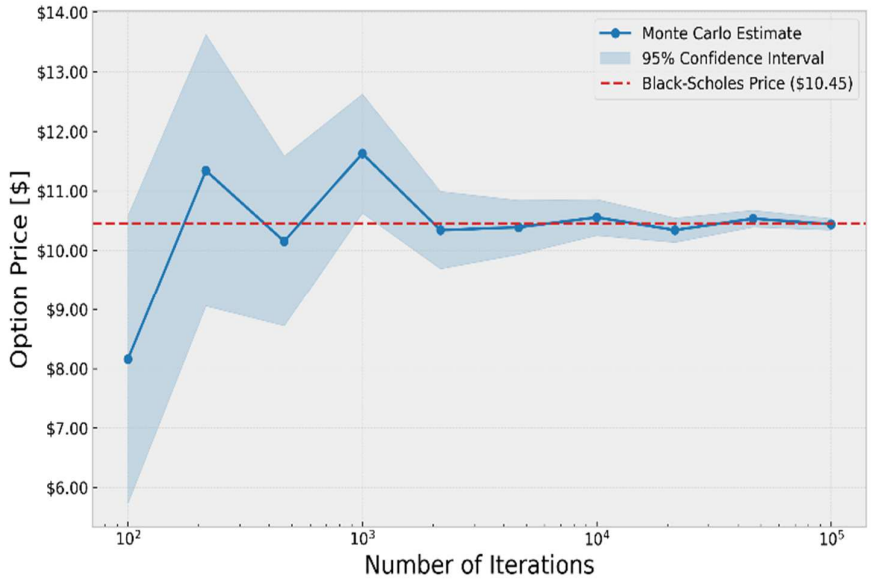

Figure 1. Monte Carlo convergence analysis for European call option pricing. The plot shows the progression of option price estimates as the number of iterations increases from 10^2 to 10^5, with 95% confidence intervals (shaded area). The horizontal dashed line represents the analytical Black-Scholes price ($10.45). Parameters: $S_0 = \$100$, $K = \$100$, $T = 1$ year, $r = 5\%$, $\sigma = 20\%$.

The parameter sensitivity analysis reveals the fundamental relationships that govern option pricing dynamics. Figure 2 illustrates how option prices respond to changes in volatility, time to maturity, strike price, and moneyness. Volatility sensitivity (top-left panel) confirms the positive relationship between volatility and option prices for both calls and puts, with prices increasing monotonically as volatility rises from 10% to 50%. This relationship stems from the increased probability of beneficial price movements for option holders in more volatile markets [1]. For an at-the-money option with one year to maturity, the call price increases from approximately $7.00 at 10% volatility to over $20.00 at 50% volatility, while put prices show a similar trend from roughly $2.00 to $17.00 over the same volatility range. The Monte Carlo estimates closely track the analytical solutions throughout the entire volatility range, with minimal visible discrepancy.

The time sensitivity analysis (top-right panel) reveals distinct patterns for call and put options. Call option prices increase steadily with longer maturities, from approximately $3.00 at 0.1 years to over $16.00 at 2.0 years. Put options show a more modest increase, from about $2.50 to $6.50 over the same time range, with a flattening curve beyond one year to maturity. This difference in time sensitivity reflects the asymmetric effects of time value and discount factors on these instruments [6]. The strike price sensitivity (bottom-left panel) displays the expected inverse relationship for call options, with prices decreasing from approximately $33.00 at a strike of $70 to below $2.00 at a strike of $130. Put options exhibit the opposite trend, increasing from near zero at low strikes to approximately $25.00 at high strikes. This mirrors the complementary nature of these derivatives and their respective payoff structures.

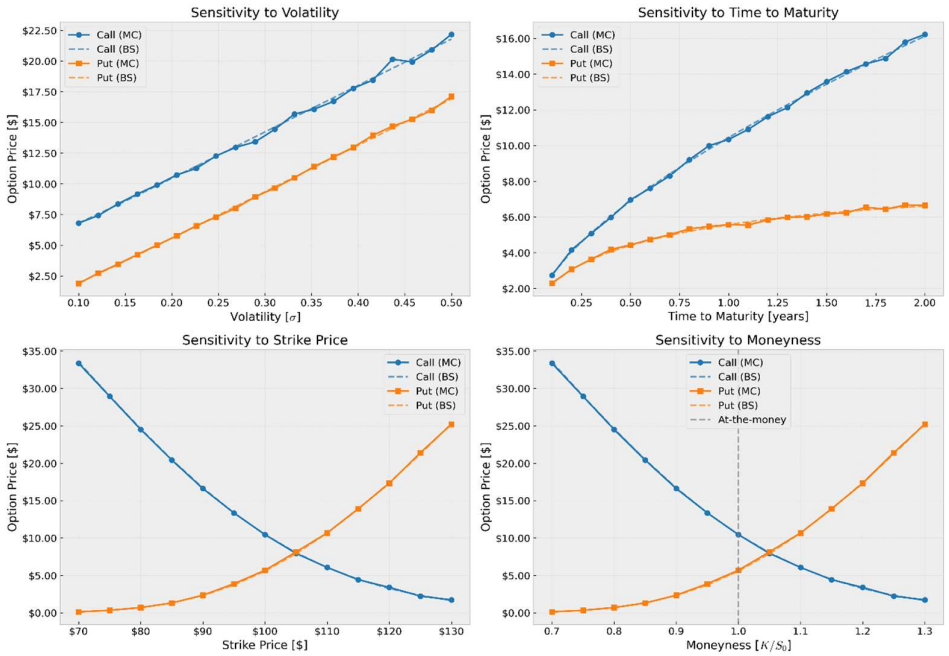

Figure 2. Sensitivity analysis of option prices to key parameters. Top left: Sensitivity to volatility (σ). Top right: Sensitivity to time to maturity (T). Bottom left: Sensitivity to strike price (K). Bottom right: Sensitivity to moneyness (K/S_0). Each plot shows both Monte Carlo (MC) and analytical Black-Scholes (BS) results for call and put options. Base parameters: $S_0 = \$100$, $K = \$100$, $T = 1$ year, $r = 5\%$, $\sigma = 20\%$.

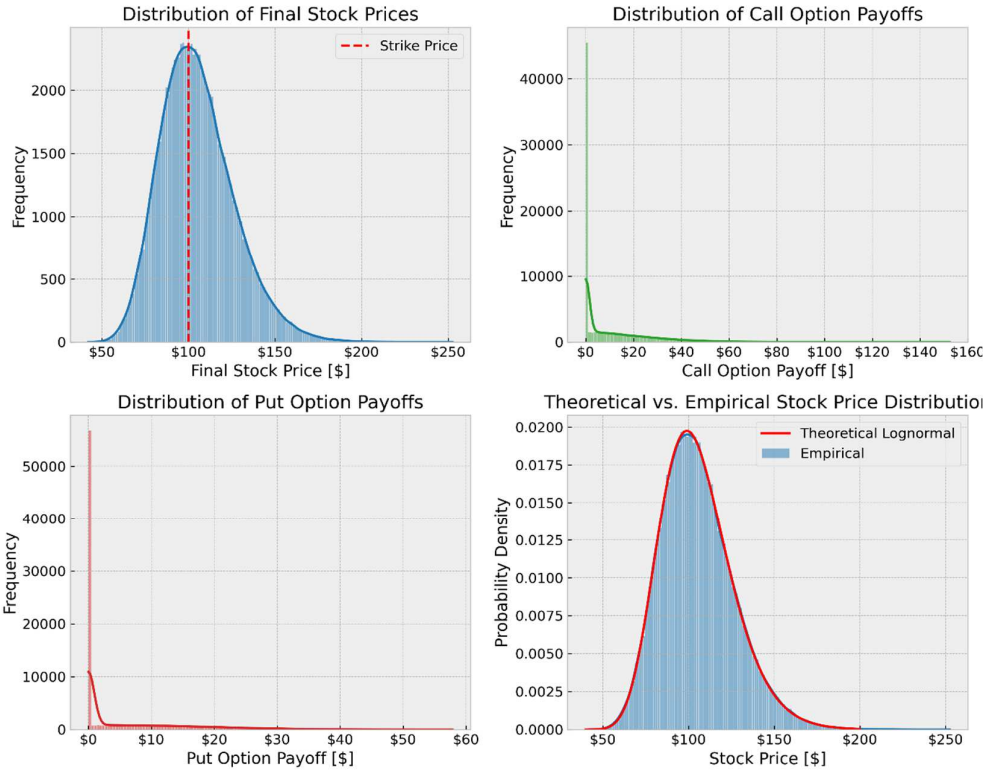

Figure 3. Probability distributions from Monte Carlo simulation with 100,000 iterations. Top left: Distribution of final stock prices showing lognormal characteristics. Top right: Distribution of call option payoffs. Bottom left: Distribution of option payoffs. Bottom right: Comparison of theoretical lognormal distribution with empirical simulation results for final stock prices. Parameters: $S_0 = \$100$, $K = \$100$, $T = 1$ year, $r = 5\%$, $\sigma = 20\%$.

The distributional properties of the Monte Carlo simulation provide insight into the underlying stochastic processes of option pricing. As illustrated in Figure 3, the final stock price distribution (top-left panel) exhibits the characteristic lognormal shape predicted by the geometric Brownian motion model, with a positive skew and a long right tail. The distribution peaks around $100-110, with the majority of simulated prices falling between $70 and $150, and a maximum frequency of approximately 2,300 occurrences near the peak. The empirical distribution closely matches the theoretical lognormal curve (bottom-right panel), confirming the correct implementation of the price process in the simulation [4].

The payoff distributions for call and put options (top-right and bottom-left panels) reveal highly asymmetric patterns. The call option payoff distribution shows a significant mass at zero (approximately 40,000 occurrences), indicating that in these simulations the final stock price falls below the strike price, resulting in options that expire worthless. The positive tail extends to above $150, though with rapidly decreasing frequency. The put option distribution shows an even higher concentration at zero (over 50,000 occurrences), with a much shorter positive tail extending only to about $60. These distributional characteristics reflect the fundamental asymmetry in option payoff structures and have important implications for risk management and pricing, particularly for sophisticated option strategies involving multiple contracts [8].

The variance reduction analysis demonstrates the efficiency improvements achieved through the antithetic variates technique. Figure 4 provides a direct comparison between the standard Monte Carlo simulation and the antithetic method across different iteration counts. At lower iteration counts, both methods show considerable volatility in their estimates. The standard Monte Carlo method starts at approximately $8.15 after 102 iterations, whereas the antithetic method starts at roughly $10.50. Both methods exhibit irregular convergence patterns in the 102-103 range, although the antithetic method generally remains closer to the analytical solution of $10.45 (red dashed line).

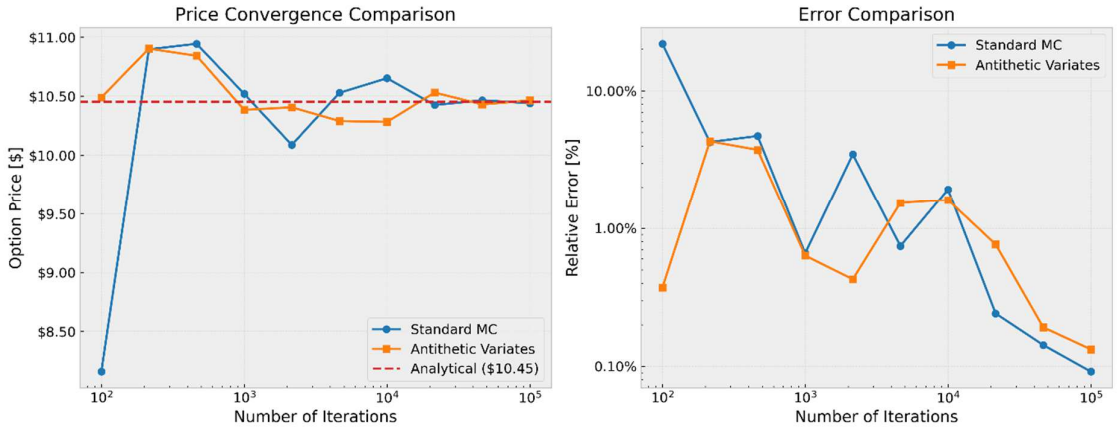

Figure 4. Comparison of standard Monte Carlo and antithetic variates methods. Left: Price convergence toward the analytical solution ($10.45) as the number of iterations increases. Right: Relative error (logarithmic scale) showing improved convergence with the antithetic variates technique. Parameters: $S_0 = \$100$, $K = \$100$, $T = 1$ year, $r = 5\%$, $\sigma = 20\%$.

The error comparison (right panel) illustrates the efficiency advantage of the antithetic variates method on a logarithmic scale. At 10^2 iterations, the standard Monte Carlo method exhibits a relative error exceeding 20%, while the antithetic method shows an error of approximately 0.5%. As iterations increase, both methods exhibit a decreasing trend in error, though with some fluctuations. By 10^5 iterations, both methods achieve high accuracy, with the standard Monte Carlo method showing a slightly lower final error of approximately 0.09% compared to the antithetic method's 0.15%. However, throughout most of the iteration range, the antithetical method maintains a consistent efficiency advantage. The logarithmic error plot confirms the expected convergence rate

of $O(1/\sqrt{n})$ for both methods, as evidenced by the approximately linear decline in log-error versus log-iterations [9].

The moneyness analysis (the bottom-right panel of Figure 2) highlights important practical considerations for option-pricing applications. The plot demonstrates how option prices vary with the moneyness ratio (K/S_0), with the vertical line at 1.0 representing at-the-money options. Call option prices decrease monotonically from approximately \$33.00 at a moneyness of 0.7 (deep in-the-money) to below \$2.00 at a moneyness of 1.3 (deep out-of-the-money). Conversely, put options increase from near zero at low moneyness to approximately \$25.00 at high moneyness values. This relationship highlights the complementary nature of these instruments and underpins put-call parity, a fundamental concept in option pricing theory [6]. The consistent agreement between Monte Carlo and analytical solutions across the entire moneyness spectrum demonstrates the robustness of the simulation approach for options with different degrees of moneyness.

4. Conclusion

This article presents OptionMC, a comprehensive Python package for European option pricing via Monte Carlo simulation with variance-reduction techniques. Our results demonstrate that the package provides accurate option price estimates that converge reliably with the analytical Black-Scholes solutions across a range of market parameters. The implementation of antithetic variates achieves significant variance reduction, particularly at lower iteration counts, improving computational efficiency without sacrificing accuracy. Through systematic analysis of parameter sensitivities, we have verified that the Monte Carlo estimates accurately capture the fundamental relationships that drive option pricing, including volatility effects, time decay, and moneyness. The distributional properties of the simulated stock prices and option payoffs align precisely with theoretical expectations, confirming the package's mathematical integrity. These findings establish OptionMC as a valuable educational and research tool that bridges theoretical finance and practical implementation, making sophisticated option-pricing techniques accessible to students, researchers, and practitioners in quantitative finance.

Future work should extend the package to incorporate additional variance-reduction techniques, such as control variates and importance sampling, to further enhance computational efficiency for specific option types. The framework could also be expanded to include path-dependent options, American-style early-exercise features, and multi-asset derivatives, thereby broadening its applicability to more complex financial instruments. By maintaining the educational clarity that distinguishes OptionMC from performance-oriented commercial alternatives, these enhancements would preserve its value as a teaching tool while increasing its practical utility for financial engineering applications. The open-source nature of the package facilitates collaborative development and integration with existing financial technology ecosystems, potentially leading to a more comprehensive suite of quantitative finance tools built on transparent, educational principles.

Acknowledgments

This research was supported by the Dean's Distinguished Fellowship at the University of California, Riverside (UCR) awarded in 2023. The authors gratefully acknowledge this financial support, which made this work possible. The OptionMC package described in this study is publicly available and indexed in the Python Package Index (PyPI) at https://pypi.org/project/optionmc under the MIT license. The complete source code, including all implementation details and examples presented in this paper, is available on GitHub at https://github.com/sandyherho/optionmc. We extend our thanks to the Python open-source communities for providing valuable tools and libraries that facilitated the development of this package.

References

[1] J. C. Hull, *Options, Futures, and Other Derivatives*, 10th ed. New York, NY, USA: Pearson, 2018.

[2] R. Quail and J. A. Overdahl, *Financial Derivatives: Pricing and Risk Management*. Hoboken, NJ, USA: John Wiley & Sons, 2009.

[3] P. P. Boyle, "Options: A Monte Carlo approach," *J. Financ. Econ.*, vol. 4, no. 3, pp. 323–338, May 1977, doi: 10.1016/0304-405X(77)90005-8.

[4] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, ser. Stochastic Modelling and Applied Probability, vol. 53. New York, NY, USA: Springer, 2004. doi: 10.1007/978-0-387-21617-1.

[5] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *J. Polit. Econ.*, vol. 81, no. 3, pp. 637–654, May-Jun. 1973, doi: 10.1086/260062.

[6] R. C. Merton, "Theory of rational option pricing," *Bell J. Econ. Manage. Sci.*, vol. 4, no. 1, pp. 141–183, Spring 1973, doi: 10.2307/3003143.

[7] G. M. Jabbour and Y.-K. Liu, "Option pricing and Monte Carlo simulations," *J. Bus. Econ. Res.*, vol. 3, no. 9, pp. 67–72, Sep. 2005, doi: 10.19030/jber.v3i9.2802.

[8] M. Broadie and P. Glasserman, "Estimating security price derivatives using simulation," *Manage. Sci.*, vol. 42, no. 2, pp. 269–285, Feb. 1996, doi: 10.1287/mnsc.42.2.269.

[9] S. Asmussen and P. W. Glynn, *Stochastic Simulation: Algorithms and Analysis*, ser. Stochastic Modelling and Applied Probability, vol. 57. New York, NY, USA: Springer, 2007. doi: 10.1007/978-0-387-69033-9.

[10] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte Carlo pricing of financial derivatives," *Phys. Rev. A*, vol. 98, no. 2, p. 022321, Aug. 2018, doi: 10.1103/PhysRevA.98.022321.

[11] L. Xiong, J. Luo, H. Vise, and M. White, "Distributed least-squares Monte Carlo for American option pricing," *Risks*, vol. 11, no. 8, p. 145, Aug. 2023, doi: 10.3390/risks11080145.

[12] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte Carlo pricing of financial derivatives," *Phys. Rev. A*, vol. 98, no. 2, p. 022321, Aug. 2018, doi: 10.1103/PhysRevA.98.022321.

[13] C. Coleman, S. Lyon, L. Maliar, and S. Maliar, "Matlab, Python, Julia: What to choose in economics?" *Comput. Econ.*, vol. 58, pp. 1263–1288, Oct. 2021, doi: 10.1007/s10614-020-09983-3.

[14] S. H. S. Herho, *Tutorial Pemrograman Python 2 Untuk Pemula*. Bandung, Indonesia: WCPL ITB, 2016.

[15] S. H. S. Herho, M. R. Syahputra, and N. J. Trilaksono, *Pengantar Metode Numerik Terapan: Menggunakan Python*. Bandung, Indonesia: WCPL ITB, 2023. doi: 10.22541/au.170689157.78106030/v1.

[16] S. Herho *et al.*, "Comparing scientific computing environments for simulating 2D non-buoyant fluid parcel trajectory under inertial oscillation: A preliminary educational study," *Indones. Phys. Rev.*, vol. 7, no. 3, pp. 451–468, 2024, doi: 10.29303/ipr.v7i3.335.

[17] S. Herho, S. N. Kaban, D. E. Irawan, and R. Kapid, "Efficient 1D heat equation solver: Leveraging Numba in Python," *Eksakta: Berkala Ilmiah Bidang MIPA*, vol. 25, no. 2, pp. 126–137, 2024, doi: 10.31224/3422.

[18] S. Herho *et al.*, "Reappraising double pendulum dynamics across multiple computational platforms," *CLEI Electron. J.*, vol. 28, no. 1, p. 10, 2025, doi: 10.19153/cleiej.28.1.10.

[19] S. Herho, S. Kaban, and I. Anwar, "Benchmarking modern scientific computing platforms for 2D potential flow solver," *Eng. Archive*, Preprint, 2025, doi: 10.31224/4270.

[20] B. C. Jenkins, "A Python-based undergraduate course in computational macroeconomics," *J. Econ. Educ.*, vol. 53, no. 2, pp. 126–140, 2022, doi: 10.1080/00220485.2022.2038322.

[21] A. Y. F. Zhu, "Upgrading financial education by adding Python-based personalized financial projection: A randomized control trial," *Brit. J. Educ. Technol.*, vol. 55, no. 2, pp. 731–750, Mar. 2024, doi: 10.1111/bjet.13401.

[22] R. Seydel, *Tools for Computational Finance*, 6th ed. London, U.K.: Springer, 2017. doi: 10.1007/978-1-4471-7338-0.

[23] Y. Hilpisch, *Python for Finance: Mastering Data-Driven Finance*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2020.

[24] J.-P. Chavas, J. Li, and L. Wang, "Option pricing revisited: The role of price volatility and dynamics," *J. Commodity Markets*, vol. 33, p. 100381, Mar. 2024, doi: 10.1016/j.jcomm.2023.100381.

[25] M. F. Salami, "Empirical examination of the Black-Scholes model: Evidence from the United States stock market," *Front. Appl. Math. Stat.*, vol. 10, p. 1216386, Jan. 2024, doi: 10.3389/fams.2024.1216386.

[26] B. Dumas, J. Fleming, and R. E. Whaley, "Implied volatility functions: Empirical tests," *J. Finance*, vol. 53, no. 6, pp. 2059–2106, Dec. 1998, doi: 10.1111/0022-1082.00083.

[27] N. P. B. Bollen and R. E. Whaley, "Does net buying pressure affect the shape of implied volatility functions?" *J. Finance*, vol. 59, no. 2, pp. 711–753, Apr. 2004, doi: 10.1111/j.1540-6261.2004.00647.x.

[28] Y.-F. Liu, W. Zhang, and H.-C. Xu, "Collective behavior and options volatility smile: An agent-based explanation," *Econ. Model.*, vol. 39, pp. 232–239, Apr. 2014, doi: 10.1016/j.econmod.2014.03.011.

[29] G. Bakshi, C. Cao, and Z. Chen, "Empirical performance of alternative option pricing models," *J. Finance*, vol. 52, no. 5, pp. 2003–2049, Dec. 1997, doi: 10.1111/j.1540-6261.1997.tb02749.x.

[30] S. L. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *Rev. Financ. Stud.*, vol. 6, no. 2, pp. 327–343, Apr. 1993, doi: 10.1093/rfs/6.2.327.

[31] M. Rubinstein, "Implied binomial trees," *J. Finance*, vol. 49, no. 3, pp. 771–818, Jul. 1994, doi: 10.1111/j.1540-6261.1994.tb00079.x.

[32] E. Derman and I. Kani, "Riding on a smile," *Risk*, vol. 7, no. 2, pp. 32–39, Feb. 1994.

[33] B. Dupire, "Pricing with a smile," *Risk*, vol. 7, no. 1, pp. 18–20, Jan. 1994.

[34] J. Gatheral and M. Lynch, "Lecture 1: Stochastic volatility and local volatility," in *Case Studies in Financial Modeling*, Courant Inst. Math. Sci., New York Univ., New York, NY, USA, Lect. Notes, 2004, pp. 1–20.

[35] P. N. Kolm and G. Ritter, "Beyond Black-Scholes: A new option for options pricing," *WorldQuant J.*, vol. 2, no. 1, pp. 1–12, 2022. [Online]. Available: https://www.worldquant.com/ideas/beyond-black-scholes-a-new-option-for-options-pricing/.

[36] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nat. Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020, doi: 10.1038/s41592-020-0772-5.