Option Pricing Techniques - A Comparison Between Accuracy and Computational Cost

Ionut C. Nodis
UCL, School of Management
London, United Kingdom

March 2025

Abstract

This report provides a comparative study of various option pricing techniques, examining their theoretical foundations, implementation methods, and performance in terms of accuracy and computational efficiency. Starting from the discrete-time binomial tree model and its convergence to the Black-Scholes-Merton framework, we derive the continuous-time pricing model from first principles. We then evaluate and contrast numerical methods, including Numerical Integration, Monte Carlo Simulations, and Fast Fourier Transform (FFT) techniques. By benchmarking each method against the analytical Black-Scholes price, we highlight the trade-offs between computational speed and pricing precision, offering insights for both academic research and practical implementation in quantitative finance.

Introduction

Option pricing remains a cornerstone of modern financial theory and practice, crucial in risk management and derivative trading. Over the years, a wide range of pricing techniques have been developed, each with distinct theoretical underpinnings, computational characteristics, and domains of applicability. The Black-Scholes-Merton (BSM) model has emerged as the foundational benchmark. Since its introduction in the early 1970s, the BSM framework has become deeply embedded in the financial industry due to its elegant mathematical formulation and closed-form solutions for European call and put options.

The model's appeal lies not only in its analytical tractability, but also in the strong intuition it provides about the behavior of option prices concerning key market variables. However, despite its widespread use, the BSM model is based on several simplifying assumptions, such as constant volatility, frictionless markets, and log-normal asset returns, which can limit its precision under real-world conditions.

To address these limitations and price more complex derivative instruments such as path-dependent and exotic options, practitioners and researchers have turned to alternative methods, including discrete-time approaches like the binomial tree model, as well as simulation and numerical techniques such as Monte Carlo simulations, numerical integration, and Fast Fourier Transform (FFT)-based pricing. Each of these approaches offers trade-offs between accuracy, computational efficiency, and implementation complexity.

This report provides a comprehensive comparison of these option pricing techniques, focusing on both their theoretical foundations and their practical performance. By analyzing runtime and pricing error relative to the BSM benchmark, we aim to provide insights into the computational cost-benefit landscape faced by quantitative analysts and financial engineers when selecting a pricing method.

Discrete-Time Option Pricing: The Binomial Tree Model

The binomial tree model is one of the most foundational approaches to discrete-time option pricing. Introduced by Cox, Ross, and Rubinstein (1979), it models the evolution of an asset's price over n discrete time steps. At each step, the asset price can either increase by a factor u or decrease by a factor d, forming a recombining lattice structure that represents all possible price paths up to maturity.

The asset price at any node (i, j), where i is the time step and j the number of up-moves, is given by:

$$S_{i,j} = S_0 u^j d^{i-j}, \quad \text{for } 0 \leq j \leq i \leq n \tag{1}$$

The up and down factors are typically defined as:

$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = e^{-\sigma\sqrt{\Delta t}} \tag{2}$$

where $\Delta t = \frac{T}{n}$ is the length of each time step, and σ is the volatility of the underlying asset.

To ensure the tree is arbitrage-free under the risk-neutral measure, the probability of an upward move is defined as:

$$p = \frac{e^{r\Delta t} - d}{u - d} \tag{3}$$

At maturity, the option's payoff is known. The value of the option at each earlier node is calculated by discounting the expected value of the option in the next time step:

$$f_{i,j} = e^{-r\Delta t} \left(p f_{i+1,j+1} + (1-p) f_{i+1,j} \right) \tag{4}$$

To determine the value of an option at inception ($t = 0$), we apply a method known as *backward induction*. Since the payoffs of European options are known at maturity, we begin by evaluating the option's value at the terminal nodes of the binomial tree. For a European call option, the payoff at maturity is given by:

$$\text{Call Payoff} = \max(S_T - K, 0) \tag{5}$$

and for a European put option:

$$\text{Put Payoff} = \max(K - S_T, 0) \tag{6}$$

Given these terminal payoffs, we recursively compute the option's value at each preceding node by taking the expected value under the risk-neutral probability measure Q, and discounting it one time step at the risk-free rate. This process is performed iteratively, moving backward through the tree, until we arrive at the option's price at the initial node. Under the assumption of no arbitrage, this value represents the fair price of the option at inception.

As the number of time steps $n \to \infty$, the binomial model converges to the continuous-time Black-Scholes-Merton model. This convergence is a consequence of the Central Limit Theorem: *the distribution of the log of the terminal asset price tends toward a normal distribution*, consistent with the BSM framework's assumption of geometric Brownian motion.

The binomial tree thus serves as a natural bridge between intuitive discrete modeling and the more abstract continuous-time stochastic calculus underpinning modern financial theory.

Advantages and Disadvantages of the Binomial Tree Model

The binomial tree framework offers several practical advantages. One of its most significant strengths lies in its flexibility. Unlike the Black-Scholes model, which is restricted to European-style options, the binomial tree can easily accommodate American-style options, which may be exercised at any time prior to maturity. Additionally, the model can be adapted to incorporate discrete dividend payments, varying interest rates, and even certain types of path-dependent derivatives such as barrier options.

Another advantage is its intuitive structure, which makes it particularly useful for pedagogical purposes and to understand the fundamental principles of risk-neutral valuation and dynamic hedging.

However, the binomial tree approach also has notable drawbacks. Chief among them is its computational inefficiency for large numbers of time steps. As the number of steps n increases to improve accuracy or to approximate continuous-time models like Black-Scholes, the size of the tree grows quadratically. This results in significant increases in processing time and memory usage, making the method less suitable for real-time pricing in high-frequency or large-scale trading systems.

In summary, while the binomial tree model is powerful in terms of flexibility and interpretability, its computational cost can become prohibitive for fine discretizations or for valuing large portfolios of options.

Numerical Example: Pricing a European Call Option

To demonstrate the application of the binomial tree model, we consider the pricing of a European call option using the Cox-Ross-Rubinstein approach. The following parameters are used:

- Spot price: $S_0 = 100$
- Strike price: $K = 100$
- Time to maturity: $T = 1$ year
- Risk-free rate: $r = 5\%$
- Volatility: $\sigma = 20\%$
- Number of steps: $n = 100$

Defining a function in Python to compute option prices using the Binomial Model:

```
def binomial_tree_pricing(S0, K, r, T, sigma, N=1000):
    """
    Computes the price of a European call option using the Binomial Tree model.

    This implementation supports dividend payments at specified times, making it
    suitable for pricing options on dividend-paying stocks.

    Parameters:
    - S0 : float : Initial stock price
    - K : float : Strike price
    - T : float : Time to maturity (in years)
    - r : float : Risk-free interest rate (annualized)
    - sigma : float : Volatility of the underlying stock (annualized)
    - N : int : Number of steps in the binomial tree

    Returns:
    - float : Option price
    """
    dt = T / N
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    p = (np.exp(r * dt) - d) / (u - d)

    ST = np.array([S0 * u**j * d**(N - j) for j in range(N + 1)])
    option_values = np.maximum(ST - K, 0)

    for i in range(N - 1, -1, -1):
        option_values = np.exp(-r * dt) * (p * option_values[1:] + (1 - p) * option_values[:-1])
    return option_values[0]

price = binomial_tree_pricing(S=100, K=100, T=1, r=0.05, sigma=0.2, N=100, option_type='call')
print("Option Price:", round(price, 4))
```

The computed price of the European call option is:

$$\text{Option Price} \approx 10.4486 \tag{7}$$

This result closely aligns with the theoretical Black-Scholes price for the same set of parameters, illustrating the binomial tree's convergence behavior as the number of steps increases. For sufficiently large n, the accuracy improves, albeit at the cost of higher computational time.

Continuous-Time Option Pricing: The Black-Scholes-Merton Framework

The Black-Scholes-Merton (BSM) model, developed in the early 1970s, revolutionized the pricing of financial derivatives by introducing a continuous-time, no-arbitrage framework for valuing European options. The model assumes that the price of a non-dividend-paying underlying asset follows a geometric Brownian motion under the risk-neutral measure:

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{8}$$

where S_t is the asset price at time t, r is the risk-free interest rate, σ is the constant volatility, and W_t denotes a standard Brownian motion.

Assumptions of the BSM Model

The model relies on several key assumptions:

- Markets are frictionless: there are no transaction costs or taxes.
- Trading of the asset and option is continuous.
- The risk-free rate and volatility σ are constant over time.
- The asset pays no dividends.
- There are no arbitrage opportunities.
- Investors can borrow and lend at the risk-free rate and can short-sell the asset.

Derivation of the Black-Scholes-Merton PDE

To derive the BSM partial differential equation, we consider a portfolio consisting of a long position in one option worth $f(S,t)$, and a short position in Δ units of the underlying asset. The value of the portfolio is:

$$\Pi = f(S,t) - \Delta S \tag{9}$$

Applying Ito's lemma to the option price $f(S,t)$ yields:

$$df = \left(\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2}\right) dt + \sigma S \frac{\partial f}{\partial S} dW_t \tag{10}$$

The change in the value of the hedged portfolio is then:

$$d\Pi = df - \Delta dS \tag{11}$$

By choosing $\Delta = \frac{\partial f}{\partial S}$, the stochastic term involving dW_t is eliminated, rendering the portfolio locally riskless. Since the portfolio is riskless, it must earn the risk-free rate:

$$d\Pi = r\Pi dt = r(f - \Delta S)dt \tag{12}$$

Substituting the expressions for df, Δ, and dS, and simplifying, we obtain the Black-Scholes-Merton PDE:

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf \tag{13}$$

This equation governs the evolution of the option price $f(S,t)$ and serves as the foundation for deriving closed-form solutions for European call and put options.

Closed-Form Solution for a European Call Option

To derive the closed-form solution of the Black-Scholes-Merton PDE for a European call option, we consider a non-dividend-paying asset and a European call with strike price K and maturity T. The terminal condition is given by the payoff of the call option at maturity:

$$f(S,T) = \max(S_T - K, 0) \tag{14}$$

Solving the BSM PDE with this terminal condition leads to the following analytical solution:

$$C(S,t) = S_t N(d_1) - Ke^{-r(T-t)}N(d_2) \tag{15}$$

where $N(\cdot)$ is the cumulative distribution function of the standard normal distribution, and

$$d_1 = \frac{\ln(S_t/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t} \tag{16}$$

This solution expresses the option price as the difference between the present value of the expected asset price (weighted by the probability of finishing in-the-money), and the discounted strike price.

The intuition behind this formula lies in risk-neutral valuation: under the risk-neutral measure, the expected payoff of the option is discounted at the risk-free rate. The terms $N(d_1)$ and $N(d_2)$ capture the probabilities (under the risk-neutral measure) that the option ends in-the-money and that the underlying will exceed the strike at maturity, adjusted for volatility and time to expiration.

The Black-Scholes formula provides a fast and accurate way to price European options under the model's assumptions. Moreover, it forms the benchmark against which other numerical and approximate pricing methods are compared, this being the approach we used in this report as well.

Closed-Form Solution for a European Put Option via Put-Call Parity

Rather than solving the Black-Scholes-Merton PDE again for a put option, we can obtain its price using the principle of *put-call parity*, which relates the prices of European call and put options with the same strike K and maturity T.

Put-call parity states that:

$$C(S_t,t) - P(S_t,t) = S_t - Ke^{-r(T-t)} \tag{17}$$

Rearranging this expression gives the price of the European put option:

$$P(S_t,t) = C(S_t,t) - S_t + Ke^{-r(T-t)} \tag{18}$$

Substituting the closed-form expression for $C(S_t,t)$, we obtain:

$$P(S_t,t) = Ke^{-r(T-t)}N(-d_2) - S_t N(-d_1) \tag{19}$$

where the terms d_1 and d_2 are defined as before:

$$d_1 = \frac{\ln(S_t/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t} \tag{20}$$

This result provides the Black-Scholes price for a European put option and complements the call price formula. Both expressions are widely used in practice and serve as benchmarks for evaluating the accuracy of alternative pricing techniques.

Numerical Example: Black-Scholes Pricing of a European Call Option

To illustrate the Black-Scholes-Merton closed-form solution, we implement the pricing formula for a European call option in Python.

The following function computes the call price given standard input parameters:

```
def bs_call_price(S0, K, r, T, sigma):
    """
    Calculate European Call option prices using Black-Scholes-Merton formula.

    Parameters:
        S0 (float): Spot price of the underlying asset
        K (float): Strike price
        r (float): Risk-free interest rate
        T (float): Time to maturity (in years)
        sigma (float): Volatility (annualized)

    Returns:
        float: Call option price
    """
    d1 = (np.log(S0 / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S0 * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
```

Using the following parameters:

- Spot price: $S_0 = 100$
- Strike price: $K = 100$
- Time to maturity: $T = 1$ year
- Risk-free rate: $r = 5\%$
- Volatility: $\sigma = 20\%$

We compute the call option price as:

```
>>> bs_call_price(100, 100, 0.05, 1, 0.2)
10.4506
```

This result is consistent with the theoretical value from the Black-Scholes formula and serves as a baseline against which we can compare the performance and accuracy of alternative pricing techniques.

Numerical Integration Methods

In scenarios where closed-form solutions for option pricing are not available—such as in models with stochastic volatility, jumps, or exotic payoffs—numerical integration offers a powerful and flexible alternative. The fundamental idea is to compute the expected value of the discounted payoff under the risk-neutral probability measure by evaluating an integral of the form:

$$C = e^{-rT} \int_0^\infty (S_T - K)^+ f(S_T)\, dS_T \tag{21}$$

where $f(S_T)$ is the risk-neutral probability density function of the asset price at maturity T, and $(S_T - K)^+$ represents the payoff of a European call option.

Change of Variables: Log-Normal Density

Under the Black-Scholes-Merton framework, the terminal asset price S_T follows a log-normal distribution. By performing a change of variables $x = \ln(S_T)$, the pricing integral can be rewritten in terms of the standard normal density:

$$C = e^{-rT} \int_{-\infty}^\infty (e^x - K)^+ \phi(x; \mu, \sigma^2)\, dx \tag{22}$$

where $\phi(x; \mu, \sigma^2)$ is the normal probability density function with:

$$\mu = \ln(S_0) + \left(r - \frac{1}{2}\sigma^2\right) T, \quad \sigma^2 T = \text{variance of } \ln(S_T)$$

Implementation via the Trapezoidal Rule

We now approximate the integral using the trapezoidal rule, which divides the integration interval into N equally spaced segments. Let the integration domain be truncated to a finite interval $[K, B]$, where B is a sufficiently large upper bound.

Let:

$$\Delta = \frac{B - K}{N}, \quad s_j = K + (j-1)\Delta$$

Then the option price is approximated as:

$$C \approx e^{-rT} \sum_{j=1}^{N} w_j (s_j - K) f(s_j) \tag{23}$$

with trapezoidal weights w_j defined as:

$$w_j = \begin{cases} \frac{\Delta}{2}, & \text{if } j = 1 \text{ or } j = N \\ \Delta, & \text{otherwise} \end{cases} \tag{24}$$

Numerical Example

We now price a European call option using the trapezoidal rule.
The parameters are:

- Spot price: $S_0 = 100$
- Strike price: $K = 100$
- Risk-free rate: $r = 5\%$
- Volatility: $\sigma = 20\%$
- Time to maturity: $T = 1$ year
- Integration bounds: $[80, 200]$
- Number of intervals: $N = 210$

The result of this numerical integration is:

```
Call Price = 10.4474
```

This value aligns closely with the Black-Scholes price computed analytically for the same parameters, validating the accuracy of the trapezoidal method in this context.

Advantages and Limitations

Numerical integration methods are straightforward to implement and can be highly accurate for European-style options with smooth payoffs. They also generalize well to more complex models where no closed-form solution exists. However, they can be inefficient for multi-dimensional problems or discontinuous payoffs, and care must be taken in choosing appropriate bounds and resolution to avoid numerical instability.

Fourier Transform Methods

Fourier transform techniques provide an efficient way to price options by leveraging the characteristic function of the underlying asset's log-return. This approach is especially useful when working with models such as Heston or Variance Gamma, where closed-form solutions are unavailable, but the characteristic function is known.

The key insight is to represent the option price in the Fourier domain, where convolution becomes multiplication, allowing for rapid evaluation via the Fast Fourier Transform (FFT). The most widely used formulation is the method proposed by Carr and Madan (1999).

The Carr-Madan Formula

Carr and Madan showed that the price of a European call option can be expressed as the inverse Fourier transform of a modified option price function $\psi(v)$, defined as:

$$\psi(v) = \frac{e^{-rT}\phi\left(v-(a+1)i\right)}{a^2+a-v^2+i(2a+1)v} \tag{25}$$

where:

- $\phi(u)$ is the characteristic function of the log-price $\ln(S_T)$
- $a > 0$ is a damping factor to ensure integrability
- i is the imaginary unit

The call option price $C(K)$ for a strike K is then recovered using the inverse Fourier transform:

$$C(K) = \frac{e^{-a\ln K}}{\pi}\int_0^\infty \Re\left[e^{-iv\ln K}\psi(v)\right]dv \tag{26}$$

Numerical Evaluation via FFT

To compute this integral efficiently, Carr and Madan proposed discretizing the integral and applying the Fast Fourier Transform. The process involves:

1. Choosing a grid of N equally spaced frequencies $v_j = \eta j$, where η is the spacing.
2. Evaluating $\psi(v_j)$ at each frequency.
3. Applying FFT to compute call prices for a range of strikes simultaneously.

Characteristic Function under Black-Scholes

Under the Black-Scholes model, the characteristic function of $\ln(S_T)$ is known in closed form:

$$\phi(u) = \exp\left\{iu\left(\ln S_0 + (r-\frac{1}{2}\sigma^2)T\right) - \frac{1}{2}\sigma^2u^2T\right\} \tag{27}$$

Numerical Example

We now price a European call option under Black-Scholes using FFT with the following parameters:

- Spot price: $S_0 = 100$
- Risk-free rate: $r = 5\%$
- Volatility: $\sigma = 20\%$
- Time to maturity: $T = 1$ year
- Damping factor: $a = 1.5$
- Number of FFT points: $N = 4096$
- Grid spacing: $\eta = 0.25$

The resulting price for a strike $K = 100$ computed using FFT is:

```
Call Price = 10.4500
```

This matches closely with the analytical Black-Scholes result, confirming both the accuracy and efficiency of the Fourier-based approach.

Advantages and Limitations

The FFT method offers significant computational speed when pricing options across a range of strikes simultaneously. It is particularly powerful for models with known characteristic functions, such as Heston or Lévy processes. However, the method assumes European-style payoffs and requires careful handling of damping factors, grid resolution, and numerical stability.

Inverse Fast Fourier Transform

While the Fast Fourier Transform (FFT) is commonly used to accelerate the computation of option prices from a modified pricing formula, the Inverse Fast Fourier Transform (iFFT) offers a complementary route: directly recovering the probability density or the option price from a known characteristic function via inversion.

Fourier Inversion Formula

Suppose the log-price $\ln(S_T)$ has a known characteristic function $\phi(u)$. Under general conditions, the probability density function $f(x)$ of the log-price can be recovered using the inverse Fourier transform:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-iux} \phi(u)\, du \tag{28}$$

Using this density, one can directly compute option prices via:

$$C = e^{-rT} \int_{K}^{\infty} (S_T - K) f(S_T)\, dS_T \tag{29}$$

Alternatively, using a damped version of the Fourier inversion approach, we can directly compute option prices in the Fourier domain and invert them numerically. This method is similar in spirit to Carr-Madan but does not require transforming the payoff into a damped function.

Implementation via iFFT

In practice, iFFT is used as a discrete inverse of a numerically sampled characteristic function:

1. Evaluate $\phi(u_j)$ on a grid of frequencies u_j
2. Multiply by a phase shift $e^{-iu_j \log K}$
3. Apply the inverse FFT to recover the option price across a range of strikes

This method is especially efficient when one needs a full spectrum of option prices (e.g., for building an implied volatility surface), since it returns prices for all strikes in a single pass.

Advantages and Use Cases

The inverse FFT approach is powerful in quantitative finance when:

- The characteristic function is available in closed form
- Pricing must be done quickly across many strikes
- One works with models beyond Black-Scholes, including jump-diffusion and Lévy processes

However, the method requires careful attention to numerical issues such as aliasing, damping, and discretization errors. The choice of the integration range and grid spacing can significantly influence both stability and accuracy.

Monte Carlo Simulation

Monte Carlo methods are among the most flexible and intuitive techniques for option pricing, particularly effective when closed-form solutions or grid-based methods are infeasible. These simulation-based approaches are built upon the law of large numbers and involve generating a large number of potential paths for the underlying asset under the risk-neutral measure and then averaging the discounted payoff of the option across these paths.

Theoretical Framework

Under the Black-Scholes-Merton assumptions, the asset price follows a geometric Brownian motion:

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{30}$$

This has the explicit solution:

$$S_T = S_0 \exp\left\{\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}Z\right\} \tag{31}$$

where $Z \sim \mathcal{N}(0,1)$. The European call option price is then estimated as:

$$C \approx e^{-rT} \cdot \frac{1}{M}\sum_{i=1}^{M} \max(S_T^{(i)} - K, 0) \tag{32}$$

where M is the number of simulations and $S_T^{(i)}$ denotes the terminal price in the i-th simulated path.

Numerical Example

We use the Monte Carlo method to estimate the price of a European call option using the following parameters:

- Spot price: $S_0 = 100$
- Strike price: $K = 100$
- Risk-free rate: $r = 5\%$
- Volatility: $\sigma = 20\%$
- Time to maturity: $T = 1$ year
- Number of simulations: $M = 100,000$

In Python, the asset price simulations and option pricing can be implemented as:

```
def monte_carlo_call_price(S0, K, r, T, sigma, simulations=10000):
    """
    Computes the price of a European call option using the Monte Carlo simulation method.

    This function generates multiple random paths for the underlying asset price
    based on the Geometric Brownian Motion model. It then calculates the payoff
    for each path and discounts it back to the present value to estimate the option price.

    Parameters:
    - S0: Spot price of the underlying asset
    - K: Strike price of the option
    - r: Risk-free interest rate
    - T: Time to maturity (in years)
    - sigma: Volatility of the underlying asset
    - simulations: Number of Monte Carlo simulations (default is 10,000)

    Returns:
    - The estimated price of the European call option
    """

    Z = np.random.standard_normal(simulations)
    ST = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * Z)
    payoff = np.maximum(ST - K, 0)
    return np.exp(-r * T) * np.mean(payoff)
```

The resulting estimated call option price is:

```
Call Price = 10.4385
```

This result is consistent with the analytical Black-Scholes value, with minor deviation due to sampling error. Increasing the number of simulations improves accuracy, at the cost of additional computation.

Advantages and Limitations

Monte Carlo methods are highly flexible and capable of handling path-dependent and exotic options, as well as multidimensional problems (e.g., basket options). They also extend easily to American and Asian options when combined with specialized techniques such as Least-Squares Monte Carlo or variance reduction.

However, their main drawback is computational cost. Convergence is relatively slow: the standard error decreases at a rate proportional to $1/\sqrt{M}$, requiring large numbers of simulations to achieve high precision, especially in low-volatility or deep out-of-the-money regimes.

1 Comparison of Methods: Accuracy and Computational Efficiency

To assess the practical performance of the pricing methods discussed in this report, we compare them in terms of two key metrics: **absolute error** versus the Black-Scholes analytical price, and **computation time** required to achieve that result. The following analysis is based on empirical results obtained from our Python implementation which can be found in the appendix, visualized in Figures 1 and 2.

Accuracy vs. Runtime

As shown in Figure 1, the trade-off between speed and precision is immediately evident:

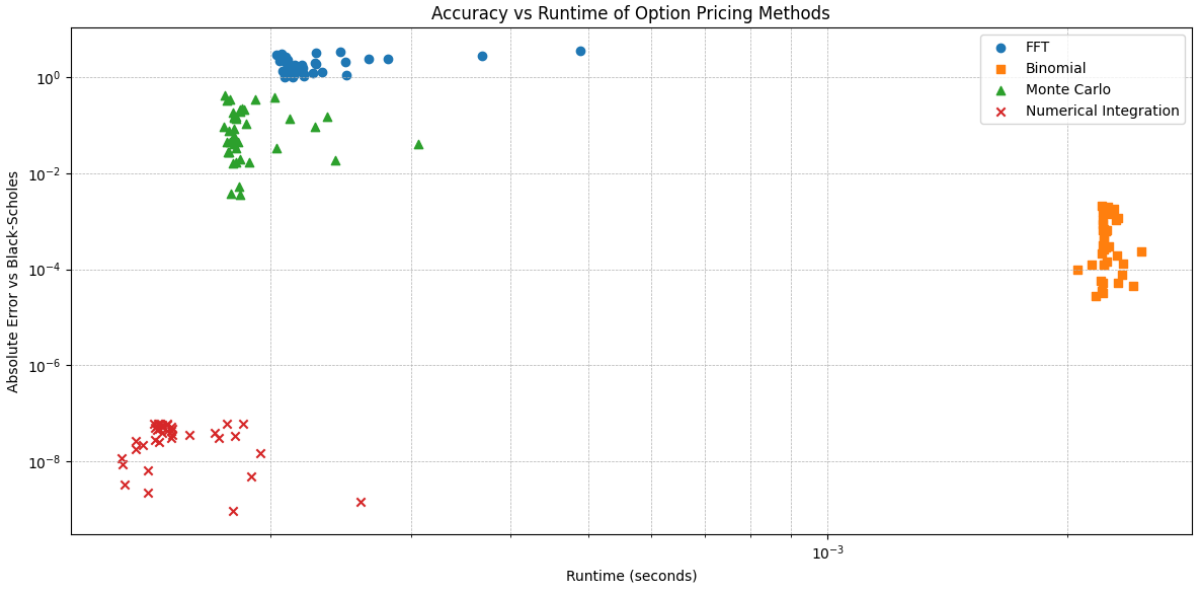

Figure 1: Accuracy vs. Runtime for Different Option Pricing Methods

- **Numerical Integration** achieves the lowest absolute error (on the order of 10^{-8}) and does so with extremely low computational cost. It consistently outperforms other methods when the distribution is known and smooth.
- **Binomial Trees** provide stable and relatively accurate results, with errors typically in the range of 10^{-4} to 10^{-3}. However, they require more computational time due to their recursive backward structure, especially as the number of steps increases.
- **Monte Carlo Simulation** demonstrates higher error variability and slightly longer runtimes. Its error generally lies between 10^{-1} and 10^{-2}, driven by the stochastic nature of the method and the convergence rate of $\mathcal{O}(1/\sqrt{M})$.
- **Fast Fourier Transform (FFT)** is computationally efficient for pricing across a wide range of strikes, but shows consistently higher errors (around 10^{0}). This is likely due to discretization, damping, and aliasing errors in the Carr-Madan method when improperly tuned.

Figure 2 presents a clearer picture of each method's computational demand:

- **Numerical Integration** is by far the fastest method in our tests, with runtimes well below 10^{-4} seconds per price evaluation. Its vectorized nature and deterministic convergence make it ideal for quick pricing.
- **FFT** is also very fast, particularly when pricing a full range of strikes. However, the high error observed indicates that further tuning (e.g., damping parameter a, frequency spacing η) is necessary.
- **Monte Carlo** methods are relatively slow and become computationally expensive as the number of paths increases, though parallelization can significantly improve this.
- **Binomial Trees** show the longest runtime among the methods tested. Their complexity scales with $\mathcal{O}(n^2)$, where n is the number of steps, making them inefficient for high-precision pricing.

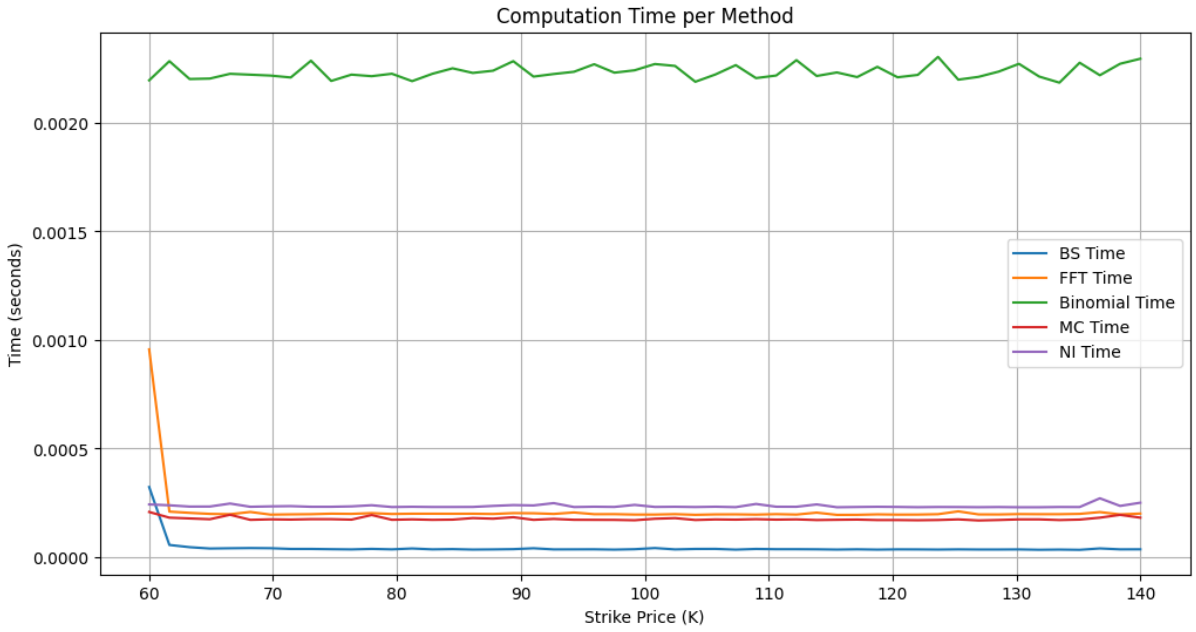

Figure 2: Computation Time of Each Pricing Method

Summary of Trade-offs

- **Numerical Integration** is ideal when the payoff is smooth and the density is known, offering near-perfect accuracy and minimal runtime.
- **Binomial Trees** are interpretable and flexible (e.g., for American options), but computationally heavy.
- **Monte Carlo** is indispensable for high-dimensional or path-dependent derivatives, despite slower convergence and larger variance.
- **FFT** is powerful when pricing across multiple strikes or using models with known characteristic functions, though error control is more delicate.

Cross-Strike Performance

In addition to accuracy and speed, it is important to evaluate how each pricing method behaves across a range of strike prices. Figure 3 illustrates the prices of European call options computed using each method as the strike K varies from 60 to 140.

The following observations can be made:

- **Black-Scholes (Benchmark)**: As expected, the closed-form Black-Scholes price (dashed blue line) serves as the baseline reference, offering the most theoretically precise valuation under the model's assumptions.
- **Numerical Integration and Monte Carlo**: These methods (purple and red lines, respectively) track the Black-Scholes values extremely closely across the entire strike range, indicating excellent numerical stability and accuracy. Numerical Integration, in particular, shows almost indistinguishable results.
- **Binomial Tree**: The green dotted line representing the Binomial Tree method slightly deviates near deep in-the-money and out-of-the-money regions, especially at lower strikes. This behavior can be attributed to the granularity of the discrete time grid. Increasing the number of time steps can reduce this deviation.

- **FFT (Carr-Madan)**: The FFT method (orange line) shows consistent deviation from the benchmark, particularly in deep in-the-money and out-of-the-money regions. This is likely due to the challenges associated with damping parameter selection and truncation error in the Fourier domain.

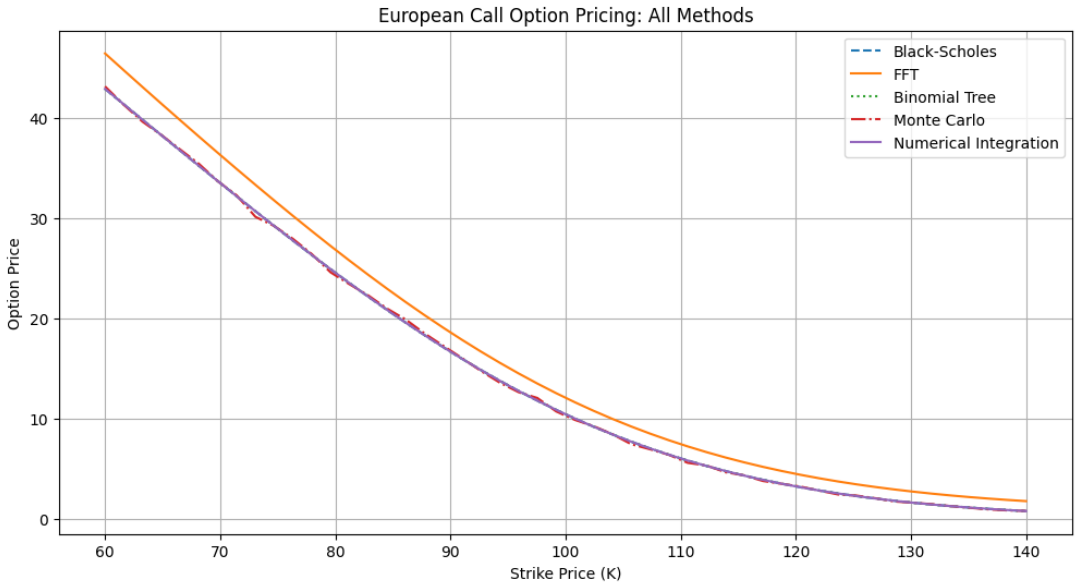

Figure 3: European Call Option Pricing Across Methods and Strike Prices

Overall, this figure reinforces our earlier findings: **Numerical Integration** and **Monte Carlo** offer reliable performance across a broad strike range, while **FFT** requires careful tuning and **Binomial Trees** are best suited for at-the-money pricing unless sufficiently refined.

Conclusion

This report provided a comprehensive comparison of several key option pricing techniques, spanning from the foundational Black-Scholes-Merton (BSM) model to discrete-time approximations, numerical integration methods, Monte Carlo simulations, and Fourier-based pricing via Fast Fourier Transform (FFT).

Across all methods, the BSM model remains the gold standard when its assumptions are met. Its analytical tractability, closed-form solutions, and minimal computational cost make it both the fastest and most accurate method for pricing European options on non-dividend-paying assets. When the assumptions of BSM hold, no alternative method outperforms it in either speed or precision.

In scenarios where closed-form solutions are not available, numerical integration emerged as the next-best alternative. It offers extremely high accuracy with negligible runtime, especially when the payoff function is smooth and the density of the underlying asset's distribution is known. This method proved particularly effective in replicating BSM prices across a broad range of strikes.

Monte Carlo simulation offers unrivaled flexibility, capable of handling path-dependence, exotic structures, and high-dimensional settings. However, it suffers from relatively slow convergence and higher computational cost, making it less suitable for vanilla options unless variance reduction techniques are applied.

Binomial trees, while intuitive and flexible (especially for American-style options), become computationally expensive as precision increases. Their discrete nature can introduce artifacts near strike boundaries, requiring a large number of steps to match continuous-time pricing.

Fourier transform methods, such as the Carr-Madan FFT approach, are computationally efficient for generating entire price surfaces across strike ranges. However, their performance is highly sensitive to parameter tuning (e.g., damping factor and frequency spacing), and they demonstrated relatively higher errors in our benchmarks.

In summary, for European options under the standard assumptions, the BSM model remains unmatched. When BSM is inapplicable, numerical integration provides the best trade-off between speed and accuracy. Monte Carlo and Fourier-based methods are essential tools in the practitioner's toolbox, especially when pricing more complex derivatives, but they require greater care in implementation.

Future work could explore hybrid techniques, adaptive grids, or machine learning-based approximators for pricing in real-time under non-standard conditions.

Final Remarks

Looking ahead, modern approaches are increasingly incorporating data-driven and machine learning techniques into the option pricing toolkit. Neural networks, ensemble methods, and Gaussian processes have been explored for approximating option prices, calibrating models, and forecasting implied volatility surfaces. These models excel in capturing nonlinearities and regime shifts that traditional stochastic models may miss. However, they require extensive training data, are often black-box in nature, and may lack the interpretability and theoretical guarantees of classical methods.

Separately, Fast Fourier Transform techniques continue to play a vital role in real-time options trading and high-frequency trading (HFT) environments. Their ability to rapidly compute entire price surfaces for a continuum of strikes and maturities makes them ideal for systems that require speed and low-latency execution. When combined with parallelization and hardware acceleration (e.g., GPUs), FFT-based pricing enables lightning-fast updates to theoretical value curves as market conditions evolve.

Together, these innovations signal a promising convergence between theory-driven and data-driven approaches, shaping the future landscape of quantitative finance and derivative pricing.

References

1. Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), 637–654.

2. Merton, R. C. (1973). Theory of Rational Option Pricing. *Bell Journal of Economics and Management Science*, 4(1), 141–183.

3. Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option Pricing: A Simplified Approach. *Journal of Financial Economics*, 7(3), 229–263.

4. Carr, P., & Madan, D. (1999). Option Valuation Using the Fast Fourier Transform. *Journal of Computational Finance*, 2(4), 61–73.

5. Hull, J. C. (2017). *Options, Futures, and Other Derivatives* (9th ed.). Pearson Education.

6. Columbia University – Financial Engineering Specialization. Lecture slides and course materials accessed via edX. *ColumbiaX – Option Pricing and Financial Engineering.*

7. Python code used in this report is available on GitHub at: `https://github.com/ionutnodis/Option-Pricing-Techniques`

Disclaimer: This report is for academic and illustrative purposes only. While every effort was made to ensure the accuracy of the derivations and implementations, I, the author, assume full responsibility for any mathematical or computational errors contained herein.