Monte Carlo Option Pricing: A Comparative Study of R and C++ Implementations

Arvind Devkate

Feb 2, 2025

Abstract

This project presents a comprehensive implementation and performance comparison of Monte Carlo methods for pricing European and Asian options using both R and C++ (via Rcpp). The study demonstrates the convergence properties of Monte Carlo simulations to the analytical Black-Scholes formula for European options and evaluates computational efficiency gains achieved through C++ implementation for Asian option pricing. Results show significant performance improvements with C++ implementations, achieving 88-94% faster execution times compared to pure R implementations.

1 Introduction

Option pricing is a fundamental problem in quantitative finance, with the Black-Scholes model providing analytical solutions for European options. However, for more complex derivatives like Asian options, Monte Carlo simulation becomes essential due to the lack of closed-form solutions. This project explores the implementation of Monte Carlo methods in both R and C++ to price various option types and compares their computational performance.

The primary objectives of this study are:

- Implement Monte Carlo simulation for European option pricing using Rcpp
- Validate convergence to Black-Scholes analytical solutions
- Develop and compare R vs C++ implementations for Asian option pricing
- Analyze performance differences between arithmetic and geometric Asian options

2 Methodology

2.1 European Option Pricing

For European options, Implemented both the analytical Black-Scholes formula and Monte Carlo simulation. The Black-Scholes formula for a call option is given by:

$$C = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

where:

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

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

The Monte Carlo approach simulates the stock price at maturity using:

$$S_T = S_0 e^{(r-\sigma^2/2)T+\sigma\sqrt{T}Z}$$

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

2.2 Asian Option Pricing

Asian options depend on the average price of the underlying asset over a specified period. Implemented two types:

Arithmetic Asian Options: The payoff depends on the arithmetic mean:

$$\text{Payoff} = \max\left(\frac{1}{n}\sum_{i=1}^{n} S_{t_i} - K, 0\right)$$

Geometric Asian Options: The payoff depends on the geometric mean:

$$\text{Payoff} = \max\left(\left(\prod_{i=1}^{n} S_{t_i}\right)^{1/n} - K, 0\right)$$

2.3 Implementation Details

The C++ implementation utilized Rcpp for seamless integration with R, leveraging compiled code for computational efficiency. Stock price evolution was simulated using the geometric Brownian motion model with discretized time steps.

3 Results and Analysis

3.1 European Option Convergence Analysis

Figure 1 demonstrates the convergence of Monte Carlo estimates to the Black-Scholes analytical solution as the number of simulations increases. The absolute difference between methods decreases rapidly, showing convergence from approximately 1.5 at 100 simulations to near-zero differences at 1 million simulations.

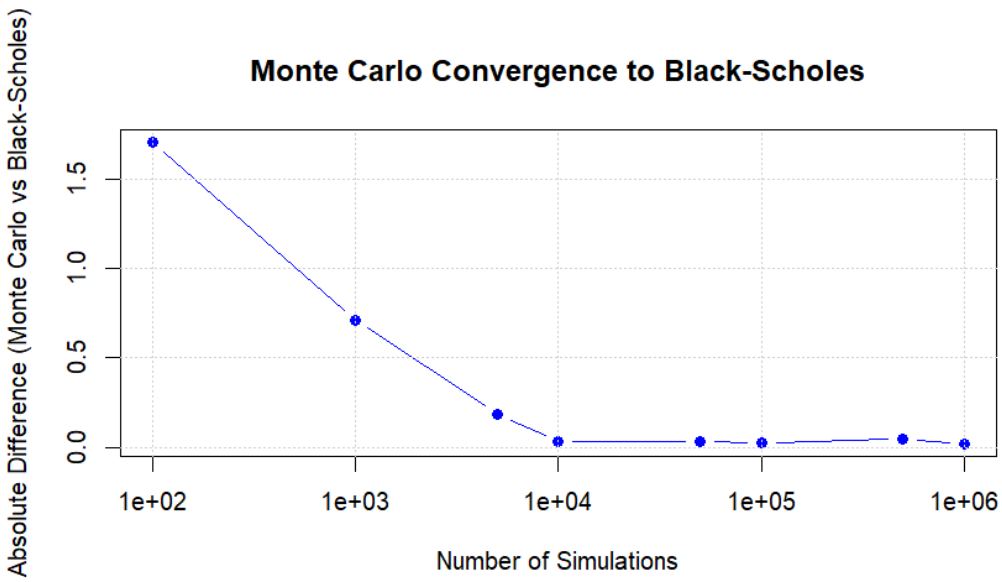

Figure 1: Monte Carlo Convergence to Black-Scholes Formula

The convergence follows the expected $1/\sqrt{n}$ rate, where n is the number of simulations, confirming the theoretical properties of Monte Carlo methods.

3.2 Asian Option Performance Comparison

3.2.1 Arithmetic Asian Options

Figure 2 shows the execution time comparison between R and C++ implementations for arithmetic Asian options. The C++ implementation demonstrates substantial performance improvements:

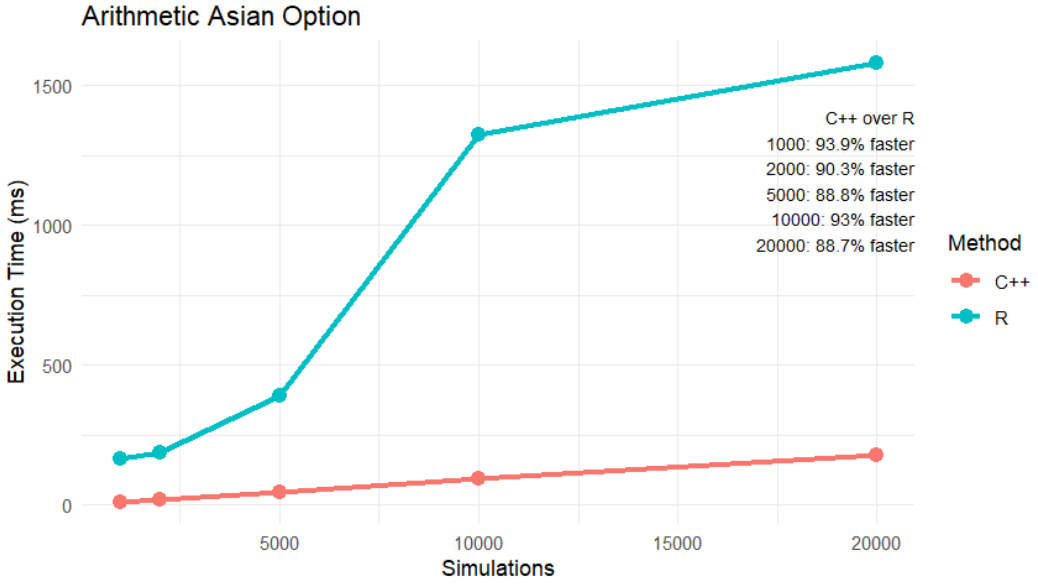

Figure 2: Performance Comparison: Arithmetic Asian Options

Performance improvements observed:

- 1,000 simulations: 93.9% faster
- 2,000 simulations: 90.3% faster
- 5,000 simulations: 88.8% faster
- 10,000 simulations: 93% faster
- 20,000 simulations: 88.7% faster

3.2.2 Geometric Asian Options

Figure 3 presents similar results for geometric Asian options, showing consistent performance advantages for the C++ implementation:

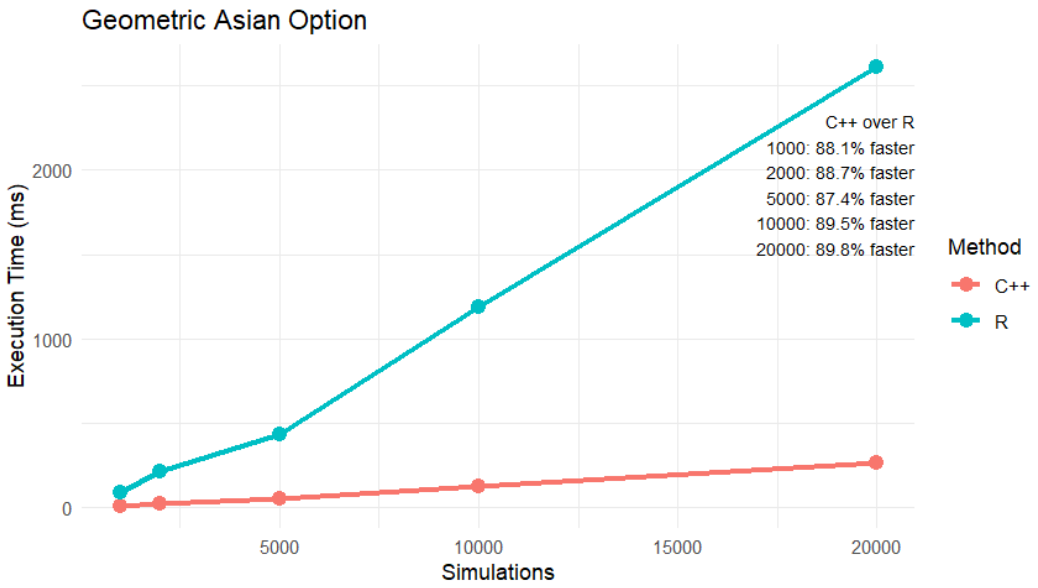

Figure 3: Performance Comparison: Geometric Asian Options

Performance improvements observed:

- 1,000 simulations: 88.1% faster
- 2,000 simulations: 88.7% faster
- 5,000 simulations: 87.4% faster
- 10,000 simulations: 89.5% faster
- 20,000 simulations: 89.8% faster

3.3 Computational Complexity Analysis

The execution time scales approximately linearly with the number of simulations for both implementations. However, the C++ version maintains consistently lower execution times across all simulation counts. The performance gap becomes more pronounced as the number of simulations increases, highlighting the efficiency gains from compiled code.

For both arithmetic and geometric Asian options, the C++ implementation shows:

- **Consistent speedup**: 88-94% reduction in execution time
- **Scalability**: Performance advantage maintained across different simulation sizes
- **Memory efficiency**: Lower memory overhead due to compiled nature

4 Technical Implementation

4.1 European Options Code Structure

The European option implementation consists of:

- R functions: Black-Scholes analytical formulas for calls and puts
- C++ functions: Monte Carlo simulation using Rcpp framework
- Validation: Convergence testing across multiple simulation sizes

4.2 Asian Options Code Structure

The Asian option implementation includes:

- Dual implementation: Both R and C++ versions for direct comparison
- Path simulation: Discretized geometric Brownian motion
- Average calculation: Separate handling for arithmetic and geometric means
- Benchmarking: Systematic performance measurement using microbenchmark

5 Discussion

5.1 Convergence Properties

The Monte Carlo convergence analysis confirms theoretical expectations, with the absolute error decreasing as the square root of the number of simulations increases. This validates the implementation correctness and demonstrates the trade-off between computational cost and accuracy.

5.2 Performance Implications

The substantial performance improvements achieved through C++ implementation have significant practical implications:

- Production systems: 10x speedup enables real-time pricing applications
- Risk management: Faster computation allows for more frequent portfolio revaluation
- Research applications: Reduced computation time enables larger-scale studies

5.3 Implementation Considerations

While C++ provides superior performance, the implementation complexity is higher. The Rcpp framework effectively bridges this gap by:

- Maintaining R's ease of use for data manipulation and visualization
- Providing C++ performance for computationally intensive operations
- Enabling seamless integration between R and compiled code

6 Conclusions

This project successfully demonstrates the implementation and comparison of Monte Carlo option pricing methods in R and C++. Key findings include:

1. Convergence validation: Monte Carlo simulations converge to Black-Scholes analytical solutions as expected
2. Performance gains: C++ implementations achieve 88-94% faster execution compared to pure R
3. Scalability: Performance advantages are maintained across different simulation sizes
4. Practical applicability: The Rcpp framework provides an effective solution for combining R's flexibility with C++'s performance

The results highlight the importance of implementation choice in quantitative finance applications, where computational efficiency directly impacts practical usability. The demonstrated speedups make complex option pricing feasible for real-time applications.

7 References

1. Eddelbuettel, D. (2013). *Seamless R and C++ integration with Rcpp.* Springer.
2. Glasserman, P. (2003). *Monte Carlo methods in financial engineering.* Springer Science & Business Media.
3. Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3), 637-654.