Option Pricing Report: Implementation and Consistency of Two Methods

Overview

This project prices European options using two methods and compares their outputs:

- Black–Scholes–Merton (BSM) closed-form solution
- Cox–Ross–Rubinstein-style Binomial Lattice (risk-neutral valuation)

Both prices and deltas are computed and displayed in *main.cpp*. The goal is to explain the implementation and evaluate whether the two methods are consistent.

Implementation Summary (Where to Look)

- Core parameters and accessors: *Option* (in *Option.h*, *Option.cpp*)
- Abstract interface: *Pricing_Method* (in *pricing_method.h*)
- Pricing engine: *Option_Price* (in *Option_Price.h*, *Option_Price.cpp*)
    - BSM price: *Option_Price::BSM_Pricer()*
    - Binomial price: *Option_Price::Binomial_Pricer()* and helper *binomialPrice(...)*
    - BSM delta: *Option_Price::BSM_Delta()*
    - Binomial delta: *Option_Price::Binomial_Delta()*
- User I/O and comparison: *main.cpp*

Black-Scholes-Merton (BSM)

Assumptions: frictionless markets, lognormal underlying with constant volatility σ, constant risk-free rate r, no dividends, and European exercise.

Implementation:

- Computes d1 and d2:
  $d1 = [\ln(S/K) + (r + 0.5\sigma^2)T] / (\sigma\sqrt{T})$, $d2 = d1 - \sigma\sqrt{T}$
- Call price: $C = S\,N(d1) - Ke^{(-rT)}\,N(d2)$
- Put price: $P = Ke^{(-rT)}\,N(-d2) - S\,N(-d1)$
- Delta: Call $\Delta = N(d1)$; Put $\Delta = N(d1) - 1$
- *normalCDF* uses the error function (erf) for numerical evaluation

These are standard, numerically stable formulas for European options under BSM assumptions.

Binomial Lattice (Risk-Neutral)

Assumptions: same economic setting as BSM but discretized in time. For a recombining tree with step size $\Delta t = T/n$:

- Up/down factors: $u = e^{(\sigma\sqrt{\Delta t})}$, $d = 1/u$
- Risk-neutral probability: $p = (e^{(r\Delta t)} - d) / (u - d)$, $q = 1 - p$
- Price is the discounted expectation of terminal payoffs:

  $V0 = e^{(-rT)} \Sigma [(n \text{ choose } j) \; p^j \; q^{(n-j)} \; \text{payoff}(S0 \; u^j \; d^{(n-j)})]$

Implementation:

- Uses n = 100 steps by default (*Option_Price::Binomial_Pricer()*)
- Computes price with a vector-free closed form using binomial coefficients in *binomialPrice(...)*
- Delta in the binomial method is computed via a finite-difference approximation from the tree's immediate up/down moves.

Expected Relationship Between Methods

For European options under BSM assumptions, the binomial lattice converges to the BSM price as $n \to \infty$. With a sufficiently large number of steps, the two methods should agree to within a small tolerance.

- Price difference decreases on the order of $1/\sqrt{n}$ to $1/n$ depending on parameterization and smoothing.
- Delta from the binomial tree converges to the BSM delta as n increases.

Practical Consistency Observations

With the default n = 100:

- Prices: For moderate parameters ($S \approx K$, $T \approx 1$, σ between 10%–60%, r between -2%–10%), the binomial price typically matches BSM within a few basis points to a

couple of cents. Smaller T, larger σ, or deep ITM/OTM cases may require larger n for tighter tolerance.

- Deltas: Binomial delta should be close to BSM delta but may be slightly noisier due to discretization. Increasing n reduces the discrepancy.

Edge-case considerations:

- Ensure the risk-neutral probability lies in [0,1]. With chosen u and d, this generally holds for reasonable Δt. Extreme parameters may require larger n.
- Very short maturity or very high volatility can magnify discretization error; increasing n improves alignment.

Conclusions: Are the Two Methods Consistent?

Yes. Under the shared assumptions for European options, the binomial lattice and BSM are theoretically consistent. Empirically, with n = 100 steps, the implementation produces prices and deltas that are close. Any residual differences come from time discretization in the binomial method, numerical evaluation of the normal CDF, and rounding. Increasing the number of steps (e.g., n = 500–1000) tightens agreement further.

In summary: the two methods are consistent; differences observed are attributable to discretization and diminish as the lattice is refined.

Recommendations

- Increase steps n for higher accuracy when parameters are extreme or tighter tolerances are required.
- Validate results with put–call parity to catch input or numerical issues.
- For Greeks from the lattice, consider a smoothed/averaged delta estimate or larger n.

How to Reproduce the Comparison

1. Compile:
   g++ -o option_pricer main.cpp Option.cpp Option_Price.cpp -std=c++11 -lm

2. Run and enter parameters (K, S, r, T, σ, option type). The program prints BSM and binomial prices and deltas, plus absolute differences.
3. To test convergence, rerun with the same inputs after modifying n in *Option_Price::Binomial_Pricer()* (e.g., 50, 100, 200, 500) and observe the differences shrink.