Comparison between Crank-Nicolson and Chebyshev: Numerical Solutions to Black-Scholes PDE

Madeline Wong, Matt Moran, Karina Gurevich, Lynne Pan, Suleman Khan, Ethan Luvisia, Notch Zhou

University of Rochester StemForAll 2025

August 10, 2025

Outline

Objectives

- Introduce the Black-Scholes Equation and the European Call/Put Market
- Compare Numerical Methods for solving the Black-Scholes PDE:
  - Analytical (Baseline)
  - Crank–Nicolson
  - Chebyshev
- Implement these methods in C++ and Python
- Identify differences and pitfalls of each method
- Conclude on accuracy and performance

Introduction

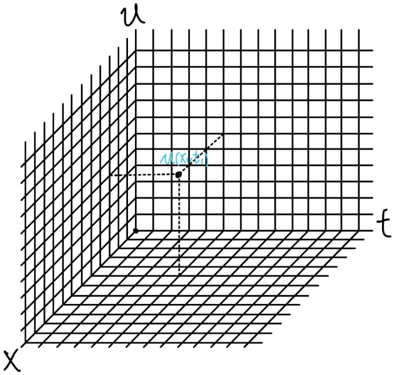
- The Black-Scholes PDE determines the option price $V(S,t)$ under:
  - No arbitrage (risk-neutral measure P^*)
  - Constant risk-free rate r
  - Known current stock price S_0
  - Log-normal return assumptions
  - Constant drift μ and volatility $\sigma > 0$

The Black-Scholes Equation

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

Crank-Nicolson Method

- Finite difference method to solve PDEs numerically
- Approximates derivatives on a grid in x and t
- Uses an implicit midpoint average between time steps
- Assumes solution is smooth enough for grid interpolation



Chebyshev Method

- Spectral method using Chebyshev polynomials for global approximation
- Evaluates PDE at Chebyshev nodes (non-uniform grid)
- Avoids Runge's phenomenon
- Highly accurate for smooth solutions

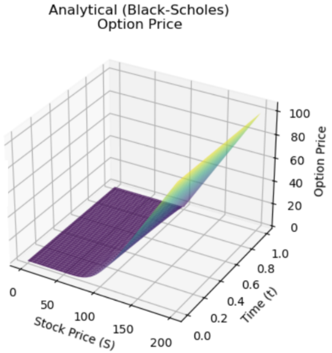
Analytical Solution

- Closed-form solution for European call option pricing
- Assumes continuous trading, constant volatility, no arbitrage
- Involves normal CDFs $N(d_1), N(d_2)$
- Formula: $C(S,t) = N(d_1) \cdot S - N(d_2) \cdot Ke^{-rT}$

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

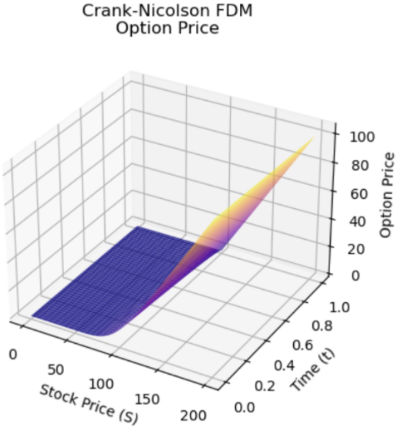
$$d_2 = d_1 - \sigma\sqrt{T}$$

$C(S,t)$	(call option price)
$N()$	(cumulative distribution function)
$T = (T_1 - t)$	(time left til maturity (in years))
S	(stock price)
K	(strike price)
r	(risk free rate)
σ	(volatility)



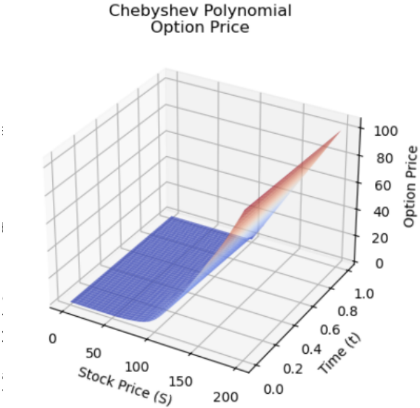
Crank-Nicolson: Evaluation

- RMSE (GitHub implementation):
    - Price: **0.107**, Delta: **0.007**, Gamma: **0.0001**
- Strong boundary behavior, handles payoff discontinuities
- Trade-off: Slower convergence, more grid refinement needed



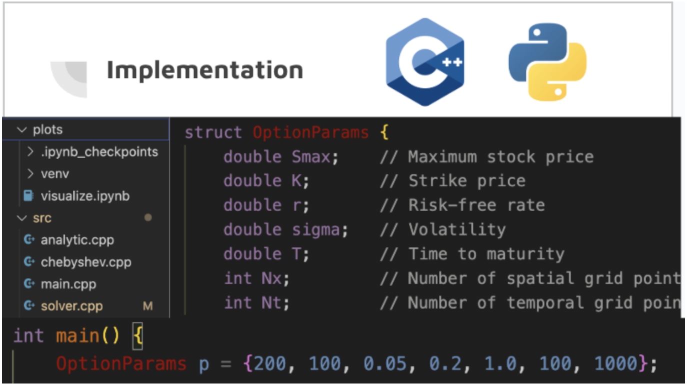
Chebyshev: Evaluation

- RMSE:
  - Price: **1.819**, Delta: **0.06**, Gamma: *undefined*
- Spectral accuracy, but sensitive to discontinuities
- Less robust near strike price and boundaries



Implementation: C++ and Python

- Core computation in C++ (option solvers, grid generation)
- Python used for visualization and analysis
- Modular design with files: `main.cpp`, `solver.cpp`, `chebyshev.cpp`, etc.



Execution Time Results

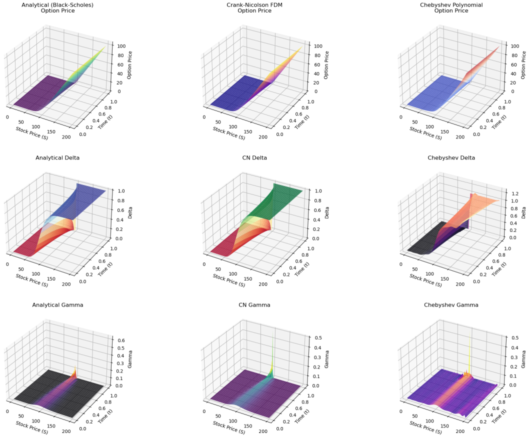
- Analytical: **91 ms**
- Crank-Nicolson: **76 ms**
- Chebyshev: **79 ms**

```
(base) Mac:build ethanmakokha$ ./option_solver
=== European Call Option Pricing Comparison ===
Parameters:
  S_max: 200, K: 100
  r: 0.05, σ: 0.2, T: 1
  Grid: 101 × 1001 points

1. Computing analytical (Black-Scholes) solutions...
   Completed in 91 ms
2. Computing Crank-Nicolson finite difference solutions...
   Completed in 76 ms
3. Computing Chebyshev polynomial approximation solutions...
   Completed in 79 ms
```

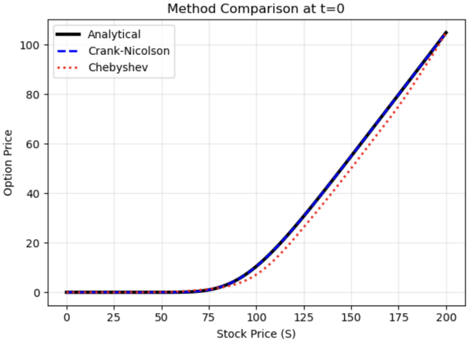
Comparison: Surface Plots

- Visual comparison of option price, delta, gamma
- Crank–Nicolson aligns best with analytic solution
- Chebyshev shows instability near kinks

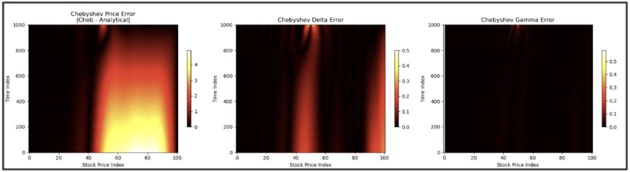


Comparison at $t = 0$

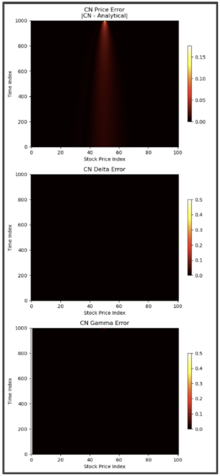
- All models compared at expiration
- Chebyshev deviates near-the-money
- Crank-Nicolson follows analytical curve closely



Error



Chebyshev Price Error
|Cheb - Analytical|
Chebyshev Delta Error
Chebyshev Gamma Error
Time Index
Stock Price Index



CN Price Error
|CN - Analytical|
CN Delta Error
CN Gamma Error
Time Index
Stock Price Index

Conclusion

- Crank-Nicolson: Robust, accurate, handles payoff kinks well
- Chebyshev: Fast and accurate in smooth regions, but unstable at discontinuities
- Future Work:
  - Refine Chebyshev boundary conditions
  - Explore adaptive volatility models

```
double ChebyshevSolver::evaluateBoundaryAwareApprox(const std::vector<double>& coeffs, do
    double x = transformToStandard(S);

    // Compute boundary term g(S, t) = linear interpolation of known boundaries
    double tau = current_tau; // set this in run() loop
    double V0 = 0.0;
    double Vmax = S - p.K * std::exp(-p.r * tau);
    double g = V0 + (Vmax - V0) * S / p.Smax;

    // Build weighted Chebyshev approximation
    double scaled = S * (p.Smax - S);
    double Tsum = 0.0;
    for (int i = 0; i < n_basis && i < (int)coeffs.size(); ++i) {
        Tsum += coeffs[i] * chebyshevBasis(i, x);
    }

    return g + scaled * Tsum;
```

References I

- Bhowmik, S.K., Khan, J.A. (2022). High-Accurate Numerical Schemes for Black–Scholes Models with Sensitivity Analysis.
- Blyth, S. (2014). *An Introduction to Quantitative Finance.*
- Caporale, G. M., Cerrato, M. Retrieved from `https://www.econstor.eu/bitstream/10419/26353/1/568606132.PDF`
- Cavendish, J.C., Culham, W.E., Varga, R.S. (2004). Comparison of Crank–Nicolson and Chebyshev Rational Methods.
- Hull, J.C. (2021). *Options, Futures, and Other Derivatives*, 11th Ed.
- Lawler, G.F. (2006). *Introduction to Stochastic Processes.*