Pricing Vanilla Options Using PDE

C++ Project

Introduction

This project aims to develop a computational framework for pricing financial derivatives using Partial Differential Equations (PDEs). The primary focus is on mono-underlying products, such as vanilla, European, and path-dependent options, leveraging the Black-Scholes model and log-normal dynamics.

By implementing:

- A robust `matrix` class, which creates matrix objects and performs inversion using the Gauss elimination method.
- A `PDEPricer` class that incorporates numerical methods for solving PDEs, allowing for the evaluation of option prices under deterministic volatility assumptions.

The main file (`main.cpp`) centralizes all operations. The results will be validated against analytical solutions from the Black-Scholes formula, emphasizing convergence and accuracy through systematic grid refinement (choices of discretization parameters `spaceSteps` and `timeSteps`). This effort bridges theoretical finance and computational efficiency, showcasing practical applications of PDEs in derivative pricing.

Results

Here are the parameters chosen to compute the price of a call option:

- **Maturity:** 1 year
- **Volatility:** 0.15
- **Interest rate:** 3%
- **Strike:** 60
- **Initial Stock Price (S_0):** 70
- **Dividend:** 0
- **Repo:** 0
- **λ:** 3

Case 1: `timeSteps` = 200, `spaceSteps` = 200

Parameter	Value	Black-Scholes Price	PDE Price
Time	337 ms	12.2798	11.9912

As we can see, the computation is quick and the result is close to the Black-Scholes price (2.4% error).

Case 2: Refining only Δt, `timeSteps` = 400, `spaceSteps` = 200

Parameter	Value	Black-Scholes Price	PDE Price
Time	543 ms	12.2798	11.9923

As we can see, increasing `timeSteps` improves the accuracy (2.3% error).

Case 3: Refining only Δx, `timeSteps` = 200, `spaceSteps` = 400

Parameter	Value	Black-Scholes Price	PDE Price
Time	1872 ms	12.2798	11.9915

As we can see, the computation is quick and the result is getting a little closer to the Black-Scholes price (2.35% error). **However, the convergence is slower than when increasing `timeSteps`.** This is coherent because increasing time Steps reduces Δt, which helps control how well the time-dependent terms (like the rate of change of the solution) are captured. Since the **pricing equation depends heavily on time-dependent changes (e.g., option decay)**, this refinement directly improves accuracy. On the other hand, **increasing `spaceSteps` refines the spatial grid, improving the approximation of spatial derivatives (e.g., delta, gamma)**. However, these terms are secondary to the time evolution of the option value, which is why the impact on convergence is less pronounced.

Case 4: Increasing Both `timeSteps` and `spaceSteps`

`timeSteps`	`spaceSteps`	Time (ms)	PDE Price
400	400	2609	11.9926
600	600	9340	11.9932
800	600	11106	11.9940
1000	600	14340	11.9949

The price converges but quite slowly compared to the increase in computation time.

Conclusion

We successfully priced a European vanilla option using a finite difference scheme based on PDE resolution. Moreover, we tested the convergence with different discretization parameters. We also saw that the speed of convergence was different depending on the nature of the changed parameters (increasing `timeSteps` gives faster convergence than increasing `spaceSteps`).

Note: Accuracy of the price comes at a cost of time since the program's complexity is quite high.