Solving Black-Scholes Equation With Finite Difference Methods

John Lain, Reese Karo, Zhenyuan Ni

March 2024

1 Introduction

The Black-Scholes equation, a cornerstone of modern financial mathematics, provides a theoretical estimate for the price of European-style options. Mathematically, it is a partial differential equation that describes the price evolution of an option over time. The equation is based on several assumptions, including constant volatility and interest rates, and it ignores dividends. Its significance lies in its ability to model the dynamics of an option's price by factoring in the underlying asset's current price, time to expiration, volatility, and the risk-free interest rate. The Black-Scholes formula revolutionized the field of financial derivatives by providing a systematic method to price options, leading to the growth of financial markets in options and other derivative securities. The main purpose of the project is to investigate the stability of numerical solutions for the Black-Scholes Equation given different initial conditions or financial settings.

A call option is a financial contract between two parties which allows the buyer of the option to buy an underlying asset for a predetermined price (called the strike price). The buyer of the call option can choose to execute the call option, meaning they choose to buy the asset at the strike price, but is not obligated to. The seller however is obligated to sell it if the buyer chooses to execute the option. Because the buyer will always have the option to execute, there is never a scenario where a rational buyer will lose money while holding the option. This means a call option will have a positive price. In our project we explore European options, which means that the buyer can only execute the option at a single future time.

If the asset price is below the strike price at the time of execution, the call option is worthless. This is because the owner of a call option would never choose to execute this option, as that would mean buying an asset for more than it's worth. However, if the asset price is above the strike price, the option is valuable as the owner will be able to buy the asset at a discount. This means that at the time of execution, a call option is worth max(asset price - strike price, 0). This gives us our "initial" condition in order to numerically solve our Black Scholes PDE.

2 Methodology

The Black-Scholes equation, which is fundamental in financial mathematics, especially for option pricing, is given by

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

where we define the variables to be V(S, t): Price of option (we will be using call options), S(t): Underlying stock Price, t: Time, σ: Volatility, r: Risk Free Interest Rate, and T: Final time/Option maturity

The finite differences we used for approximating the first and second derivatives are as such:

$$\frac{\partial V}{\partial t}(S,t) \approx \frac{1}{k}(V(S,t+k) - V(S,t)) \tag{2}$$

$$\frac{\partial V}{\partial S}(S,t) \approx \frac{1}{2h}(V(S+h,t) - V(S-h,t)) \tag{3}$$

$$\frac{\partial^2 V}{\partial S^2}(S,t) \approx \frac{1}{h^2}(V(S+h,t) - 2V(S,t) + V(S-h,t)) \tag{4}$$

Where one can replace k with Δt and $2h$ and h^2 with $2\Delta S$, ΔS^2 respectively if they prefer different notation.

In the numerical approximation, we use centered difference for the second order term, as well as the centered difference for the first derivative with respect to the stock price. The above setup would produce an explicit solution in the case of certain PDE problems, such as the heat equation we explored in class. However, due to the context of option pricing, we actually know the price distribution at the end our our time interval, giving the following initial conditions and boundary conditions:

$$V(0,t) = 0 \text{ for all t}$$
$$V(S,t) \approx S - Ke^{-r(T-t)} \text{ as S} \to \infty$$
$$V(S,T) = \max(S-K,0)$$

Which then leads us to the introduction of the backwards difference quotient for the first derivative with respect to time:

$$V_t(S,t) \approx \frac{1}{k}(V(S,t) - V(S,t-k) \tag{5}$$

Our reason for doing this is that this backwards difference quotient can be used to create an explicit method in our context. This is because our initial condition

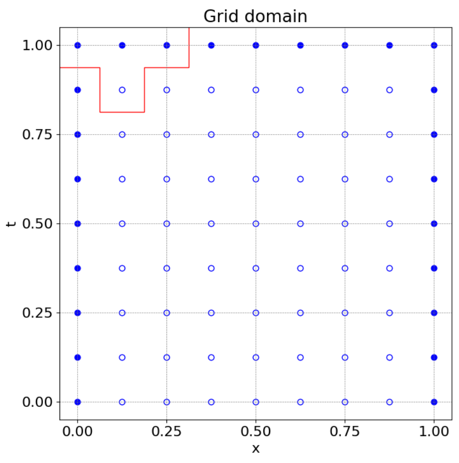


Figure 1: Black Scholes PDE Stencil, showing how we use the time point above to calculate

is actually evaluated at some future time, rather than at time zero. So, in each step we compute, we iterate backward through time, while having a solution for the future time. Justification for this can be found here. The reader can also view the stencil in figure 1

3 Computational Setup

The computational setup involves defining a grid over the domain of interest and marching through time to solve the equation at each grid point. The Boundary conditions:

$$C(0,t) = 0 \text{ for all } t \tag{6}$$

$$C(S,t) \sim S - Ke^{-r(T-t)} \text{ as } S \to \infty \tag{7}$$

$$C(S,T) = \max\{S-K, 0\} \tag{8}$$

where K is the strike price, volatility defined as σ and the risk-free rate as R are specified. The spatial domain is defined from the left boundary, A to right boundary, B, and the temporal domain is from initial time 0 to the final marching time, T . The number of spatial grid points, N and time grid points, K are chosen based on the desired resolution. We want to make note that the notation has changed from V(S,t) which represents the value of an ***option***, to C(S,t) which only represents the value of a ***call*** option. Again the call option gives the buyer the right, but not the obligation, to buy stock at certain price and a certain time in the future. The step size in the stock price domain, ΔS and time domain, ΔT

By replacing Eqn.(1) with (2),(3), and (4), we end up with the following discretization using backward finite difference for the time derivative,

$$\frac{1}{k}\left(w_{i,j}-w_{i,j-1}\right)+S_i^2\frac{\sigma^2}{2h^2}\left(w_{i+1,j}-2w_{ij}+w_{i-1,j}\right)+rS_i\frac{1}{2h}\left(w_{i+1,j}-w_{i-1,j}\right)-rw_{i,j}=0 \tag{9}$$

Then with some simple algebra techniques, solving for $w_{i,j+1}$ and separating the spatial steps,

$$\begin{aligned} w_{i,j-1} &= \left(\frac{k}{2h^2}S_i^2\sigma^2 - rS_i\frac{k}{2h}\right)w_{i-1,j} \\ &+ \left(1 - kr - \frac{k}{h^2}\sigma^2 S_i^2\right)w_{i,j} \\ &+ \left(\frac{k}{2h^2}\sigma^2 S_i^2 + \frac{k}{2h}rS_i\right)w_{i+1,j} \end{aligned} \tag{10}$$

Then we can define the following for simpler notation $\delta_i := \frac{k}{2h^2}\sigma^2 S_i^2$, $\rho := kr$, $\phi_i := \frac{k}{2h}rS_i$. Thus with another strategy of simple algebraic arrangement, we end up with the system in matrix form.

$$\begin{bmatrix} w_{1,j-1} \\ \vdots \\ w_{m,j-1} \end{bmatrix} = \begin{bmatrix} 1-\rho-2\delta_1 & \delta_1+\phi_1 & 0 & \cdots & 0 \\ \delta_2-\phi_2 & 1-\rho-2\delta_2 & \delta_2+\phi_2 & \ddots & \vdots \\ 0 & \delta_3-\phi_3 & 1-\rho-2\delta_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \delta_{m-1}+\phi_{m-1} \\ 0 & \cdots & 0 & \delta_m-\phi_m & 1-\rho-2\delta_m \end{bmatrix} \begin{bmatrix} w_{1j} \\ \vdots \\ w_{mj} \end{bmatrix} + \begin{bmatrix} (\delta_1-\phi_1)w_{0,j} \\ 0 \\ \vdots \\ 0 \\ (\delta_m+\phi_m)w_{m+1,j} \end{bmatrix}$$

This allows us to visualize a linear representation of the problem, which can either be solved by updating the option price iteratively via (10), or in matrix form, we can use matrix solvers.

4 Code

In the block of code shown below, we see the ability to solve numerically for the next step spatially by updating our vector w throughout each spatial point, at time j-1. We do this by following 10. Though not implemented, can be changed to be solved by a matrix solver discussed above.

```python
import numpy as np

```

```python
def solve_fdm_option_backwards(w, par):
    """
    Return the next time of the option price using backwards
    difference method (but this is an explicit method).

    Input:
        w: (1D array) Option price distribution at current.
        par: (dict) Parameter for the method.
    Output:
        w_new: (1D array) Option price distribution one time step
    backwards.
    """
    w_new = np.zeros_like(w[1:-1])

    # Main line of the difference method described above:
    w_new = (1 -par['k']*par['r'])*w[1:-1] \
        + par['S']*par['S']*par['k']*(par['sig']*par['sig'])/(2*(
    par['h']*par['h']))*(w[2:] - 2.*w[1:-1] + w[:-2]) \
        + par['k']*par['r']* par['S']*1/(2*par['h'])*(w[2:] - w
    [:-2])

    return w_new
```

Listing 1: Solving option prices using backwards finite difference in the time variable

5 Results

In our testing, we define error as follows, in order to have a single value to reference: We look at our time 0 distribution of prices, and find the absolute value of the difference between the numerical solution and the real solution for each spatial step. Then, we find the average of these errors across all or our spatial steps. Looking at only time zero results is limiting, however in context it can be thought of as the fair price to pay today for the option. Notice how our graph is very close to the real solution. The average error at time zero was about 0.003, which in the context of option pricing would be less than half a cent.

After graphing a solution for a single set of values, we chose a set of 8 spatial steps and 8 time steps to test. This means we tested 64 different conditions and for the error for each one. In figure 3, the rows represent changes in number of spatial steps and the columns represent changes in number of time steps. We can see that as the number of spatial steps increases, the error decreases, although somewhat inconsistently. We can also see that this effect also occurs for an increase in the number of time steps, but less significantly. However, at 50 steps, the error increases slightly for time steps, showing that error may have been slightly amplified. Another interesting result is that when using 500 steps and 200 spatial steps, we had an error of 0.0002, lower than all values in our table. It appears that stability is conditional on both spatial and temporal step size, and that increasing the number of special steps requires an increase in time steps to keep a stable solution. Our method has an error of $O(h^2 + k)$ due to

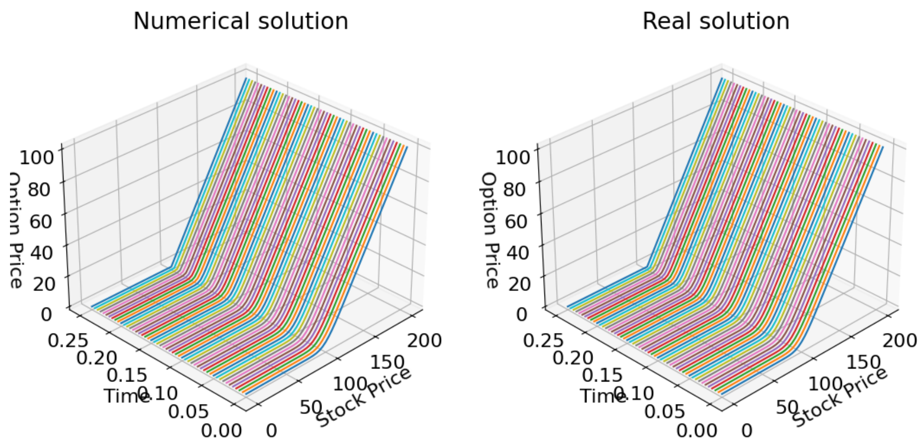
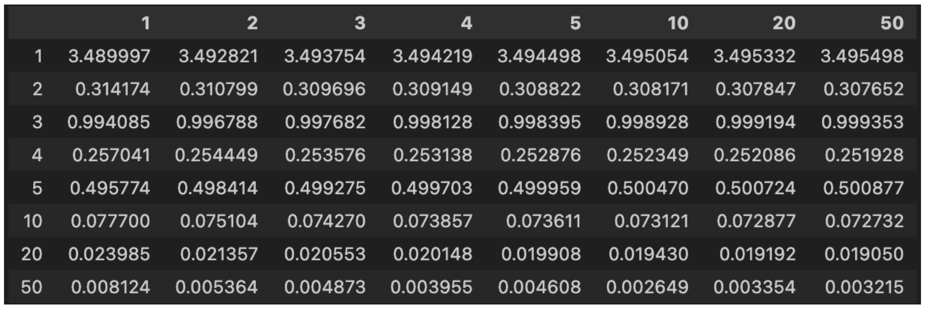


Figure 2: Black-Scholes Computational Results



	1	2	3	4	5	10	20	50
1	3.489997	3.492821	3.493754	3.494219	3.494498	3.495054	3.495332	3.495498
2	0.314174	0.310799	0.309696	0.309149	0.308822	0.308171	0.307847	0.307652
3	0.994085	0.996788	0.997682	0.998128	0.998395	0.998928	0.999194	0.999353
4	0.257041	0.254449	0.253576	0.253138	0.252876	0.252349	0.252086	0.251928
5	0.495774	0.498414	0.499275	0.499703	0.499959	0.500470	0.500724	0.500877
10	0.077700	0.075104	0.074270	0.073857	0.073611	0.073121	0.072877	0.072732
20	0.023985	0.021357	0.020553	0.020148	0.019908	0.019430	0.019192	0.019050
50	0.008124	0.005364	0.004873	0.003955	0.004608	0.002649	0.003354	0.003215

Figure 3: Error Rates for Different Time and Step Sizes

an analysis which is similar to those done in close, where h is spatial step size and k is time step size. So, our result in implementation of the method is the expected one.

6 Stability Analysis

When setting all parameters constant, increasing the number of spatial steps past a certain point introduces instability into our solution. In our code, 50 spatial steps created a visually stable solution, however 100 spatial steps did not. We set T to 0.25 in our project, as higher values often created unstable solutions. In the context of option pricing, this value of T corresponds to a quarter of a year, or 3 months, which is a reasonable time till expiration of an option. When looking at our method from the perspective of a fixed point iteration, finding the eigenvalues of the matrix which is multiplied by our current conditions is a difficult task. This is because the matrix is not Toeplitz, as each row is dependent on a different spatial grid point. So, finding eigenvalues requires more detail than discussed in this course so far. Because of this, we did not find conditions which create stability.

7 Conclusion

The successful implementation of a finite difference method with minimal error in finance shows its power for solving questions about options that do not have explicit solutions. In our analysis of errors, the faster convergence of the numerical solution to the real solution when spatial step size decreases compared to a decrease in time step size demonstrates the theoretical difference that we know. It is important to note that in practice however, increasing our number of spatial steps (and thus decreasing the size of these steps) and holding all else constant actually had a negative effect on the accuracy of our numerical solution after a certain point due to instability. An analysis of stability could be done by someone with the tools to solve for the eigenvalues of the matrix we encountered.