Financial Modelling

Pricing Exotic Equity Derivatives by Solving the Black Scholes Partial Differential Equation With Finite Difference Methods

JOSHUA FORDAY
NICHOLAS KWOK

Word Count: 998

Imperial College London

15/05/22

1 Introduction

Financial instruments such as derivatives enable investors to hedge risk, profit and speculate on asset classes. A derivative is a contract and derives its value from an underlying asset. A basic derivative is the European call option, which gives the purchaser the right, not the obligation, to buy a stock at price K, at maturity time, T.

Exotic derivatives are complex instruments where an investment bank would price and structure this product depending on the client. These instruments could contain a basket of contracts which depend on multiple assets.

In this Computing II coursework, we will explore how a type of exotic derivative, a two-colour rainbow option, is priced, by using the two-asset Black Scholes Partial Differential Equation.

2 Black Scholes Partial Differential Equation

The two-colour rainbow option consists of two underlying equities, S_1 and S_2, correlated via constant ρ. τ is defined as $\tau = T - t$, t is time and T is time to maturity, a constant. Dependent variable, u, denotes the price of the option and is a function of independent variables S_1, S_2, and τ. Constants σ_1 and σ_2 are the volatilities of S_1 and S_2 respectively and r is the risk-free interest rate.

The parabolic PDE models $u(S_1,\ S_2,\ \tau)$:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma_1 S_1^2 \frac{\partial^2 u}{\partial S_1^2} + \frac{1}{2}\sigma_2 S_2^2 \frac{\partial^2 u}{\partial S_2^2} + \rho\sigma_1\sigma_2 S_1 S_2 \frac{\partial^2 u}{\partial S_1 \partial S_2} + rS_1 \frac{\partial u}{\partial S_1} + rS_2 \frac{\partial u}{\partial S_2} - ru \tag{1}$$

The partial differential equation is solved in a bounded finite domain of $[(0,\ 200)\times (0,\ 200)\times\ (0,\ T)]$. 200 is the top bound for the two equities.

3 Boundary and Initial Conditions

The boundary conditions are:

$$\left.\frac{\partial^2 u}{\partial S_1^2}\right|_{S_1=0} = \left.\frac{\partial^2 u}{\partial S_1^2}\right|_{S_1=\ 200} = \left.\frac{\partial^2 u}{\partial S_2^2}\right|_{S_2=0} = \left.\frac{\partial^2 u}{\partial S_2^2}\right|_{S_1=200} = 0 \tag{2}$$

These conditions imply that the price of the equities is linear with the price of the option at the boundaries - a characteristic of European options [1].

The initial condition is when τ=0, and thus t=T, which represents the payoff of the option at maturity, T. Different options strategies can be structured

and have different payoffs. A butterfly spread option consists of longing two call options at strike price K_1, K_2, and shorting a call option at strike price $K = (K_1 + K_2)/2$, [2]. The payoff is:

$$u\left(S_1,\ S_2,\ \tau = 0\right) = \max\left(s - K_1,\ 0\right) - 2\max\left(s - K,\ 0\right) + \max\left(s - K_2,\ 0\right) \quad (3)$$

The butterfly spread option describes a market neutral strategy with capped profit and fixed risk. More strategies are coded, but this assignment will focus on the butterfly strategy.

4 Numerical Method

The numerical method used to solve Equation 1 is the **Alternating Directions Implicit Method (ADI)**. Explicit methods could be used, however, these methods take time to converge and have small regions of stability. Additionally, Crank Nicolson could be used, but is computationally expensive as it requires populating and solving a matrix with complicated systems of equations, which takes memory [3].

The ADI method is memory efficient as it generates a tridiagonal matrix which can be solved using the Thomas Algorithm (TDMA). It converges quicker than both explicit, Crank Nicolson methods and is 2nd order accurate.

5 Discretising the Partial Differential Equation

For clarity, let $\mathcal{L}_{BS}$ be defined as an operator:

$$\mathcal{L}_{BS} = \frac{1}{2}\sigma_1 S_1^2 \frac{\partial^2 u}{\partial S_1^2} + \frac{1}{2}\sigma_2 S_2^2 \frac{\partial^2 u}{\partial S_2^2} + \rho\sigma_1\sigma_2 S_1 S_2 \frac{\partial^2 u}{\partial S_1 \partial S_2} + rS_1 \frac{\partial u}{\partial S_1} + rS_2 \frac{\partial u}{\partial S_2} - ru \quad (4)$$

Where:

$$\mathcal{L}_{BS} = \frac{\partial u\left(S_1,\ S_2,\ \tau\right)}{\partial \tau} \quad (5)$$

The ADI Method is semi-implicit [4], and progresses forward in 2 stages of half steps, with the first half step in the S_1 domain taken implicitly and the other S_2 domain taken explicitly. In second half step S_2 is taken implicitly and S_1 is taken explicitly. i is defined as ranging from $0,\ \ldots N_x$, where N_x is the number of intervals in the S_1 domain, j is similarly defined from $0,\ \ldots N_y$, discretising the S_2 domain, and n discretises the τ domain. Equations 6 and 7 describe the 2 stages of the ADI method.

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\delta\tau} = \mathcal{L}_{BS}^{S_1}\, u_{ij}^{n+\frac{1}{2}} \tag{6}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\delta\tau} = \mathcal{L}_{BS}^{S_2}\, u_{ij}^{n+1} \tag{7}$$

Combining:

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\delta\tau} = \mathcal{L}_{BS}^{S_1}\, u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{BS}^{S_2}\, u_{ij}^{n+1} \tag{8}$$

Using finite difference approximations, $\mathcal{L}_{BS}^{S_1}$ and $\mathcal{L}_{BS}^{S_2}$ were discretised, leading to Equations 9 and 10. h is defined as the step size in the S_1 and S_2 domain.

$$\begin{aligned}\mathcal{L}_{BS}^{S_1} u_{ij}^{n+\frac{1}{2}} = \frac{(\sigma_1 S_{1,i})^2}{4}\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{(\sigma_2 S_{2,j})^2}{4}\frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{h^2} \\ + \frac{1}{2}\rho\sigma_1\sigma_2 S_{1,i} S_{2,j}\frac{u_{i+1,j+1}^{n} + u_{i-1,j-1}^{n} - u_{i-1,j+1}^{n} - u_{i+1,j-1}^{n}}{4h^2} \\ + \frac{1}{2} r S_{1,i}\frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j}^{n+\frac{1}{2}}}{h} + \frac{1}{2} r S_{2,j}\frac{u_{i,j+1}^{n} - u_{i,j}^{n}}{h} - \frac{1}{2} r u_{ij}^{n+\frac{1}{2}}\end{aligned} \tag{9}$$

$$\begin{aligned}\mathcal{L}_{BS}^{S_2} u_{ij}^{n+1} = \frac{(\sigma_1 S_{1,i})^2}{4}\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{(\sigma_2 S_{2,j})^2}{4}\frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{h^2} \\ + \frac{1}{2}\rho\sigma_1\sigma_2 S_{1,i} S_{2,j}\frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{i-1,j-1}^{n+\frac{1}{2}} - u_{i-1,j+1}^{n+\frac{1}{2}} - u_{i+1,j-1}^{n+\frac{1}{2}}}{4h^2} \\ + \frac{1}{2} r S_{1,i}\frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j}^{n+\frac{1}{2}}}{h} + \frac{1}{2} r S_{2,j}\frac{u_{i,j+1}^{n+1} - u_{i,j}^{n+1}}{h} - \frac{1}{2} r u_{ij}^{n+1}\end{aligned} \tag{10}$$

To solve for the first half step $u_{i,j}^{n+\frac{1}{2}}$, Equation 6 is rearranged with the discretised operator $\mathcal{L}_{BS}^{S_1}$ (Equation 9) substituted. A tridiagonal matrix, A_{s_1}, can be formed and the system is described.

$$A_{S_1} u_{0:N_x,j}^{n+\frac{1}{2}} = f_{0:N_x,j} \tag{11}$$

A_{s_1} is as follows:

$$A_{S_1} = \begin{bmatrix} 2\alpha_0 + \beta_0 & \gamma_0 - \alpha_0 & 0 & \cdots & 0 & 0 \\ \alpha_1 & \beta_1 & \gamma_1 & \cdots & 0 & 0 \\ 0 & \alpha_2 & \beta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & 0 & 0 & \cdots & \alpha_{N_x} - \gamma_{N_x} & \beta_{N_x} + 2\gamma_{N_x} \end{bmatrix} \tag{12}$$

Coefficients α_i, β_i, γ_i, and f_{ij} are defined as:

$$\alpha_i = -\frac{(\sigma_1 S_{1,\,i})^2}{4h^2} \tag{13}$$

$$\beta_i = \frac{1}{\delta\tau} + \frac{(\sigma_1 S_{1,\,i})^2}{2h^2} + \frac{r S_{1,\,i}}{2h} + \frac{r}{2} \tag{14}$$

$$\gamma_i = -\frac{(\sigma_1 S_{1,\,i})^2}{4h^2} - \frac{r S_{1,i}}{2h} \tag{15}$$

$$\begin{aligned} f_{ij} = \frac{u^n_{ij}}{\delta\tau} - \frac{(\sigma_2 S_{2,j})^2}{4}\frac{u^n_{i,\,j+1} - 2u^n_{ij} + u^n_{i,j-1}}{h^2} + \frac{1}{2} r S_{2,\,j}\frac{u^n_{i,j+1} - u^n_{i,j}}{h} \\ + \frac{1}{2}\rho\sigma_1\sigma_2 S_{1,i} S_{2,j}\frac{u^n_{i+1,\,j+1} + u^n_{i-1,j-1} - u^n_{i-1,\,j+1} - u^n_{i+1,j-1}}{4h^2} \end{aligned} \tag{16}$$

The system A_{s_1} and f_{ij} can be solved using TDMA, which removes the need to store a matrix. The algorithm is shown.

```python
import numpy as np

def TDMA(ld, md, td, arr): #Arguments are the diagonals
    '''
    In this case, ld is alpha, md is beta, td is gamma, and arr is
    fij or gij

    '''
    n = len(md) #Length of the main diagonal
    q = 0.0
    sol = np.zeros(n, dtype = float) #Initialise solution array

    #Copy array in order to avoid changing the original tridiagonal
     matrix

    lower = np.copy(ld)
    middle = np.copy(md)
    top = np.copy(td)
    array = np.copy(arr)

    #Elimination step (removes sub-diagonal elements, alpha)

    for k in range(1, n):

        q = lower[k]/middle[k-1]

        middle[k] = middle[k] - top[k-1]*q

        array[k] = array[k] - array[k-1]*q

    q += array[n-1]/middle[n-1]

    sol[n-1] = q

```

```
    #Back Substitution and solves

    for k in range(n-2, -1, -1):

        q = (array[k] - top[k]*q)/middle[k]

        sol[k] = q

    return sol #Outputs the solution to the linear system of
    equations
```

Listing 1: TDMA example

A similar process is conducted for the next half time step, with A_{S_2} identical in form to A_{S_1} with new coefficients α_j, β_j, γ_j and a new g_{ij}, defined below.

$$A_{S_2} u^{n+1}_{i,\ 0:N_y} = g_{i,\ 0:N_y} \tag{17}$$

$$\alpha_j = -\frac{(\sigma_2 S_{2,\ j})^2}{4h^2} \tag{18}$$

$$\beta_j = \frac{1}{\delta\tau} + \frac{(\sigma_2 S_{2,\ j})^2}{2h^2} + \frac{r S_{2,\ j}}{2h} + \frac{r}{2} \tag{19}$$

$$\gamma_j = -\frac{(\sigma_2 S_{2,\ j})^2}{4h^2} - \frac{r S_{2,j}}{2h} \tag{20}$$

$$\begin{aligned} g_{ij} = \frac{u_{ij}^{n+\frac{1}{2}}}{\delta\tau} + \frac{(\sigma_1 S_{1,\ i})^2}{4} \frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{1}{2} r S_{1,i} \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j}^{n+\frac{1}{2}}}{h} \\ + \frac{1}{2}\rho\sigma_1\sigma_2 S_{1,i} S_{2,j} \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{i-1,j-1}^{n+\frac{1}{2}} - u_{i-1,j+1}^{n+\frac{1}{2}} - u_{i+1,j-1}^{n+\frac{1}{2}}}{4h^2} \end{aligned} \tag{21}$$

The matrix was solved using TDMA, after conducting both half steps, the partial differential equation is solved for one time step, $\delta\tau$. This process iterates from $\tau = 0,\ \delta\tau,\ 2\delta\tau,\ \ldots,\ N_\tau$.

A further detail is evaluating points at the edges of the domain. Certain indices were not possible and thus the Neuman boundary condition was used to rewrite the inaccessible index. For example, at $j = 0$ the bold terms needed to be reevaluated:

$$\begin{aligned} f_{ij} = \frac{u_{ij}^n}{\delta\tau} - \frac{(\sigma_2 S_{2,\ j})^2}{4} \frac{u_{i,\ j+1}^n - 2u_{ij}^n + \boldsymbol{u_{i,j-1}^n}}{h^2} + \frac{1}{2} r S_{2,\ j} \frac{u_{i,\ j+1}^n - u_{i,j}^n}{h} \\ + \frac{1}{2}\rho\sigma_1\sigma_2 S_{1,\ i} S_{2,\ j} \frac{u_{i+1,\ j+1}^n + \boldsymbol{u_{i-1,\ j-1}^n} - u_{i-1,\ j+1}^n - \boldsymbol{u_{i+1,\ j-1}^n}}{4h^2} \end{aligned} \tag{22}$$

To rewrite $u^n_{i,\ j-1}$, the 2nd order central difference approximation was used at $j = 0$ and solved for $u_{i,\ j-1}$:

$$\left.\frac{\partial^2 u}{\partial S_2^2}\right|_{S_2=0} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0 \tag{23}$$

$$u_{i,j-1} = 2u_{i,j} - u_{i,j+1} \tag{24}$$

This process was used for all other inaccessible points, including the corners of the domain.

6 Numerical Experiments and Plots

Figure 1 shows the surface plot of a price of the option at time $\tau = T$ and as a function of S_1 and S_2.

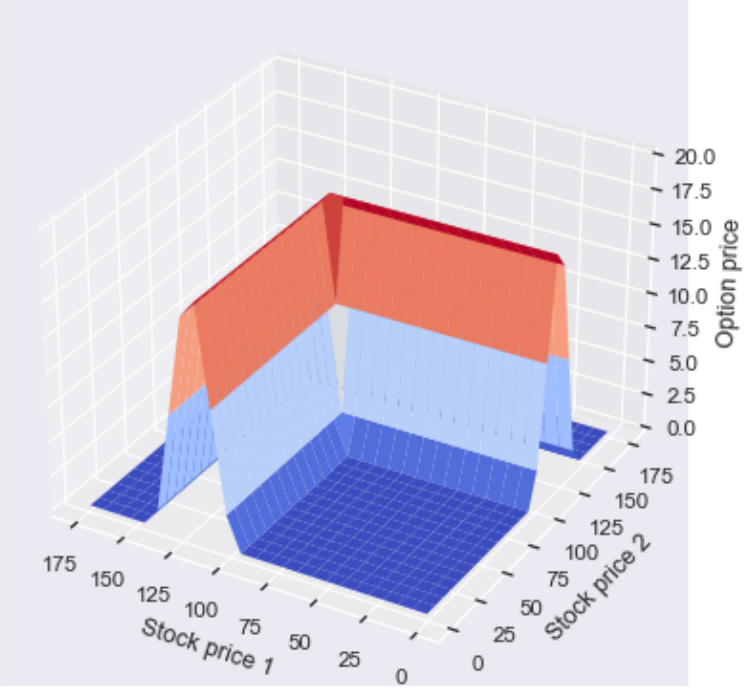

Figure 1: Option Price at Maturity, $\tau = T$

As τ increases, in the regions where the option is "in the money" (ITM), the price decreases as shown by Figure 2. ITM is when exercising the option is favourable.

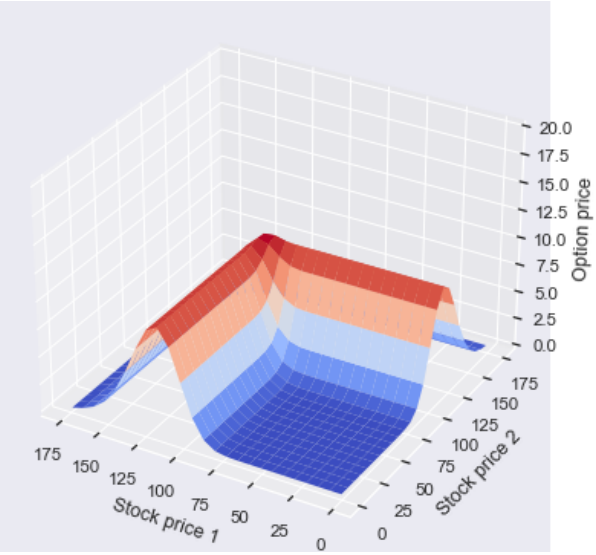

Figure 2: Option Price at $\tau > 0$

This is in line with intuition, as an ITM option will be most expensive close to maturity, as there is a greater certainty of profit.

By testing with finer domains the Figures 3, 4 were produced.

These graphs show the blow-up phenomenon when the space step size is too small and is a common disadvantage of the ADI method.

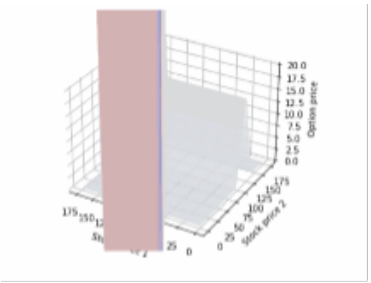

Figure 3: 10x finer grid.

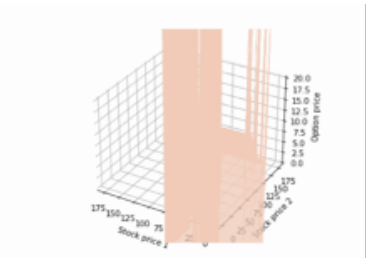

Figure 4: 50x finer grid.

7 Remarks

The semi-implicit nature of the ADI method causes the solution to blow up if the space steps are too small, shown in Figures 4 and 5. This is a known issue with the ADI method and so a current field of research is to find more robust methods.

One such developing solution is the Operator Splitting (OS) method. Like the ADI method it progresses the solution in 2 stages. However, it crucially solves both S1 and S2 implicitly in both stages making it unconditionally stable while having similar errors. Thus, the solution does not blow up and is convergent at small steps. Despite this, the ADI method shows better convergence with large steps [2] and so the method selected depends on the requirements.

8 References

[1] J. Persson, L. von Sydow, Pricing European multi-asset options using a space–time adaptive FD-method, Comput. Vis. Sci. 10 (2007) 173–183.

[2] Darae, J. Junseok, K. (2022) A comparison study of ADI and operator splitting methods on option pricing models. Journal of Computational and Applied Mathematics. 247, 162–171.

[3] Douglas, J. Kim, S. (1999) On Accuracy of Alternating Direction Implicit Methods for Parabolic Equations.

[4] Radoslav L, V. (2015) Numerical Solution of a Two-asset Option Valuation PDE by ADI Finite Difference Discretization. American Institute of Physics.

[5] Geng, Weihua Zhao, Shan. (2013). Fully implicit ADI schemes for solving the nonlinear Poisson-Boltzmann equation. Molecular Based Mathematical Biology [electronic only]. 1. 10.2478/mlbmb-2013-0006.