Advanced Option Theory Coursework

[redacted]
E. Peterson - [redacted]
[redacted]

May 23, 2024

Question 1

(a) Derive the PDE for the Digital Option

Using arbitrage pricing theory, the value of the option V must satisfy the following stochastic differential equation (SDE):

$$dV = \frac{\partial V}{\partial t} dt + \frac{\partial V}{\partial X} dX + \frac{1}{2} \frac{\partial^2 V}{\partial X^2} (dX)^2.$$

Substituting $(dX)^2 = \sigma^2 X^2 dt$:

$$dV = \frac{\partial V}{\partial t} dt + \frac{\partial V}{\partial X} dX + \frac{1}{2} \sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} dt.$$

We construct an arbitrage portfolio $P = hX + V$ and note that:

$$dP = h\, dX + dV.$$

Substituting dV:

$$dP = h\, dX + \left(\frac{\partial V}{\partial t} dt + \frac{\partial V}{\partial X} dX + \frac{1}{2} \sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} dt \right).$$

This simplifies to:

$$dP = \left(\frac{\partial V}{\partial X} + h \right) dX + \frac{1}{2} \sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} dt + \frac{\partial V}{\partial t} dt.$$

The only term that involves risk is the term $\left(\frac{\partial V}{\partial X} + h \right) dX$. To completely eliminate the risk of the hedge portfolio P, we choose the weight h on X such that:

$$\frac{\partial V}{\partial X} + h = 0 \quad \Rightarrow \quad h = -\frac{\partial V}{\partial X}.$$

Then the hedge portfolio will bear no risk, and by no arbitrage, it will earn the competitive risk-free interest rate r, which is constant in our setup. So for $h = -\frac{\partial V}{\partial X}$, we get:

$$dP + h\delta X dt = \frac{1}{2} \sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} dt + \frac{\partial V}{\partial t} dt + h\delta X dt = rP\, dt.$$

Substituting $P = -\frac{\partial V}{\partial X} X + V$, we have:

$$\frac{1}{2} \sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} dt + \frac{\partial V}{\partial t} dt - \frac{\partial V}{\partial X} \delta X dt = r \left(-\frac{\partial V}{\partial X} X + V \right) dt$$

Rearranging terms and factoring out dt, we get:

$$\frac{1}{2}\sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} + (r-\delta)X\frac{\partial V}{\partial X} - rV + \frac{\partial V}{\partial t} = 0.$$

Thus, the partial differential equation (PDE) for the digital option is:

$$\frac{1}{2}\sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} + (r-\delta)X\frac{\partial V}{\partial X} - rV + \frac{\partial V}{\partial t} = 0.$$

(b) Laplace Transform Solution

The option has a payoff at maturity ($\tau = 0$) given by:

$$V(X,0) = \begin{cases} 1 & \text{if } X \le E, \\ 0 & \text{if } X > E, \end{cases}$$

where E is the exercise price.

The value of the option $V(X,\tau)$ at time $\tau = T - t$ is governed by the PDE:

$$\frac{\partial V}{\partial \tau} + \frac{1}{2}\sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} + (r-\delta)X\frac{\partial V}{\partial X} - rV = 0$$

To solve this using the Laplace transform, let $h(X)$ be the Laplace transform of $V(X,\tau)$:

$$h(x) = \mathcal{L}_q\{V(X,\tau)\} = \int_0^\infty V(X,\tau)e^{-q\tau}\,d\tau$$

Using the Laplace transform properties, we can transform the PDE into:

$$qh(X) - V(X,0) + \frac{1}{2}\sigma^2 X^2 h''(X) + (r-\delta)Xh'(X) - rh(X) = 0$$

Given the boundary conditions:

- When $X \ge E$, $V(X,0) = 0$
- When $X < E$, $V(X,0) = 1$

We need to solve the following PDEs in the Laplace domain:

Case 1: When $X \le E$

$$\frac{1}{2}\sigma^2 X^2 h''(X) + (r-\delta)Xh'(X) - (r+q)h(X) = -1$$

Case 2: When $X > E$

$$\frac{1}{2}\sigma^2 X^2 h''(X) + (r-\delta)Xh'(X) - (r+q)h(X) = 0$$

Non-Homogeneous Solution

For the non-homogeneous PDE (Case 1), we guess the form $h(X) = b$:

$$b = \frac{1}{r+q}$$

General Solution

The general solutions are of the form:

Case 1: When $X \leq E$

$$h(X) = A_{11}X^{\gamma_1} + A_{12}X^{\gamma_2} + \frac{1}{r+q}$$

Case 2: When $X > E$

$$h(X) = A_{21}X^{\gamma_1} + A_{22}X^{\gamma_2}$$

Boundary Conditions and Matching

Using boundary conditions and matching conditions, we find:

1. As $X \to \infty$, $V(X, \tau) \to 0$, implying $h(\infty) = 0$. Hence, $A_{21} = 0$.
2. As $X \to 0$, $V(0, \tau) = e^{-r\tau}$, implying $h(0) = \frac{1}{r+q}$. Since γ_2 is negative, $A_{12} = 0$.

So, we have:

$$h(X) = \begin{cases} A_{11}X^{\gamma_1} + \frac{1}{r+q}, & \text{if } X \leq E \\ A_{22}X^{\gamma_2}, & \text{if } X > E \end{cases}$$

Matching Conditions at $X = E$

From the value matching condition:

$$A_{22}E^{\gamma_2} = A_{11}E^{\gamma_1} + \frac{1}{r+q}$$

From the smooth pasting condition:

$$A_{22}\gamma_2 E^{\gamma_2 - 1} = A_{11}\gamma_1 E^{\gamma_1 - 1}$$

Solving these, we get:

$$A_{11} = \frac{E^{-\gamma_1}}{(r+q)(\gamma_1 - 1)}$$

$$A_{22} = \frac{E^{-\gamma_2}\gamma_1}{(r+q)(\gamma_1 - 1)}$$

$$h(X) = \begin{cases} A_{22}X^{\gamma_2}, & \text{for } X > E \\ A_{11}X^{\gamma_1} + \frac{1}{r+q}, & \text{for } X \leq E \end{cases}$$

These are the expressions for $h(X)$ in the respective regions, validating the solution to the Laplace transform of the PDE governing the digital option's price.

(c) Greeks and the PDE

(i) Expressing the PDE in terms of Greeks

Recall the PDE:

$$\frac{\partial V}{\partial \tau} + (r-\delta)X\frac{\partial V}{\partial X} + \frac{1}{2}\sigma^2 X^2 \frac{\partial^2 V}{\partial X^2} = rV$$

We substitute the Greeks into the PDE:

$$\frac{\partial V}{\partial \tau} = -\Theta$$

$$\frac{\partial V}{\partial X} = \Delta$$

$$\frac{\partial^2 V}{\partial X^2} = \Gamma$$

So the PDE becomes:

$$-\Theta + (r-\delta)X\Delta + \frac{1}{2}\sigma^2 X^2 \Gamma = rV$$

(ii) Verifying the given relation

We need to verify the given relation for Θ:

$$\Theta = rV + e^{-r\tau}\phi(d)\left(r - \delta - \frac{1}{2}\sigma^2 - \frac{d}{2\tau}\right)$$

where $\phi(d)$ is the standard normal density function:

$$\phi(d) = \frac{1}{\sqrt{2\pi}}e^{-\frac{d^2}{2}}$$

Using the given option price:

$$V(X_\tau, \tau) = e^{-r\tau}\left(1 - \mathcal{N}(d)\right)$$

We find the Greeks - Delta (Δ):

$$\Delta = \frac{\partial V}{\partial X} = e^{-r\tau}\frac{\partial}{\partial X}\left(1 - \mathcal{N}(d)\right) = -e^{-r\tau}\frac{\partial \mathcal{N}(d)}{\partial X} = -e^{-r\tau}\phi(d)\frac{\partial d}{\partial X}$$

Since $\frac{\partial d}{\partial X} = \frac{1}{X\sigma\sqrt{\tau}}$:

$$\Delta = -e^{-r\tau}\phi(d)\frac{1}{X\sigma\sqrt{\tau}}$$

Gamma (Γ):

$$\Gamma = \frac{\partial^2 V}{\partial X^2} = \frac{\partial \Delta}{\partial X} = \frac{\partial}{\partial X}\left(-e^{-r\tau}\phi(d)\frac{1}{X\sigma\sqrt{\tau}}\right)$$

Using the product rule:

$$\Gamma = -e^{-r\tau}\left[\frac{\partial}{\partial X}\left(\phi(d)\right)\frac{1}{X\sigma\sqrt{\tau}} + \phi(d)\frac{\partial}{\partial X}\left(\frac{1}{X\sigma\sqrt{\tau}}\right)\right]$$

Applying the chain rule, we have:

$$\frac{\partial \phi(d)}{\partial X} = -d\phi(d)\frac{1}{X\sigma\sqrt{\tau}}$$

and

$$\frac{\partial}{\partial X}\left(\frac{1}{X\sigma\sqrt{\tau}}\right) = -\frac{1}{X^2\sigma\sqrt{\tau}}$$

Therefore:

$$\Gamma = e^{-r\tau}\phi(d)\left(\frac{d}{X^2\sigma^2\tau} - \frac{1}{X^2\sigma\sqrt{\tau}}\right)$$

Simplifying:

$$\Gamma = e^{-r\tau}\phi(d)\frac{1}{X^2\sigma^2\tau}(d-1)$$

Theta (Θ) is defined as the partial derivative of the option price with respect to time t, but here we use $\tau = T - t$:

$$\Theta = -\frac{\partial V}{\partial \tau}$$

Using the option price formula:

$$V(X_\tau, \tau) = e^{-r\tau}\left(1 - \mathcal{N}(d)\right)$$

we differentiate V with respect to τ:

$$\Theta = -\frac{\partial V}{\partial \tau}$$

Applying the product rule:

$$\frac{\partial V}{\partial \tau} = \frac{\partial}{\partial \tau}\left(e^{-r\tau}\right)\left(1 - \mathcal{N}(d)\right) + e^{-r\tau}\frac{\partial}{\partial \tau}\left(1 - \mathcal{N}(d)\right)$$

Differentiating $e^{-r\tau}$:

$$\frac{\partial}{\partial \tau}\left(e^{-r\tau}\right) = -re^{-r\tau}$$

and

$$\frac{\partial}{\partial \tau}\left(1 - \mathcal{N}(d)\right) = -\phi(d)\frac{\partial d}{\partial \tau}$$

Now, $\frac{\partial d}{\partial \tau}$:

$$d = \frac{\log\left(\frac{X}{E}\right) + \left(r - \delta - \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}$$

Differentiating d with respect to τ:

$$\frac{\partial d}{\partial \tau} = \frac{\left(r - \delta - \frac{\sigma^2}{2}\right)\sigma\sqrt{\tau} - \left(\log\left(\frac{X}{E}\right) + \left(r - \delta - \frac{\sigma^2}{2}\right)\tau\right)\frac{\sigma}{2\tau^{1/2}}}{\sigma^2\tau}$$

Simplifying, we get:

$$\frac{\partial d}{\partial \tau} = \left(\left(\frac{r - \delta - \frac{\sigma^2}{2}}{\sigma\sqrt{\tau}}\right) - \frac{d}{2\tau}\right)$$

Combining these results, we have:

$$\Theta = re^{-r\tau}(1-\mathcal{N}(d)) + e^{-r\tau}\phi(d)\left(\left(\frac{r-\delta-\frac{\sigma^2}{2}}{\sigma\sqrt{\tau}}\right) - \frac{d}{2\tau}\right)$$

Rearranging and simplifying:

$$\Theta = rV + e^{-r\tau}\phi(d)\left(\frac{r-\delta-\frac{\sigma^2}{2}}{\sigma\sqrt{\tau}} - \frac{d}{2\tau}\right)$$

This verifies the given relation for Θ:

$$\Theta = rV + e^{-r\tau}\phi(d)\left(\frac{r-\delta-\frac{\sigma^2}{2}}{\sigma\sqrt{\tau}} - \frac{d}{2\tau}\right)$$

Question 2

(a) Numerical Solution using Explicit Finite Difference Scheme

$V(S_t, t)$ is the value of an up-and-out barrier European Call option written on a stock, whose price is denoted by S_t. The dynamics of the stock price are given by:

$$dS_t = \mu S_t dt + \sigma(t, S_t) S_t dW_t \tag{1}$$

The volatility surface has been calibrated with the following functional form:

$$\sigma(t, S_t) = 0.13e^{-t}(\frac{100}{S_t})^{\alpha} \tag{2}$$

The pricing equation of the barrier option is similar to that of its Vanilla equivalent i.e. it satisfies the Black-Scholes PDE:

$$\frac{\partial V}{\partial t} + \frac{\sigma(t,S_t)^2}{2} S_t^2 \frac{\partial^2 V}{\partial S^2} + rS_t\frac{\partial V}{\partial S} - rV = 0 \tag{3}$$

with terminal condition:

$$V_t = V(S_T, T) = \begin{cases} \max(S_t - K, 0), & \text{if } \max\limits_{0\le t\le T} S_t < D, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We use the parameters defined in Table 1 to solve the PDE numerically, by means of an explicit finite difference scheme.

S_0	K	r_f	D	T (years)	α	P
\$100	\$100	1.8%	\$118	0.5	0.24	\$2.96

Table 1: Up-and-out European Barrier Call, Parameters and Price

See Figure 1 for the price surface and Figure 2 for the finite differences mesh of the up-and-out barrier call.

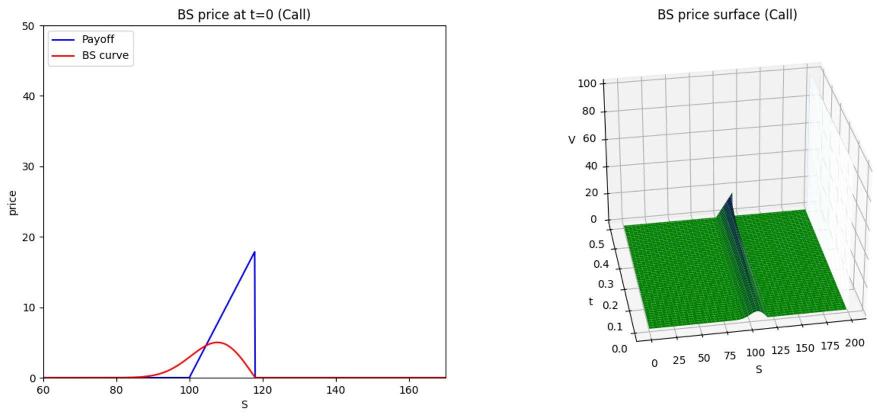


Figure 1: European Up-and-out Barrier Call Option (Table 1) Payoff and Price Vs. Underlying (left), V(t, S) - Price Surface, t - Time, S - Underlying (right)

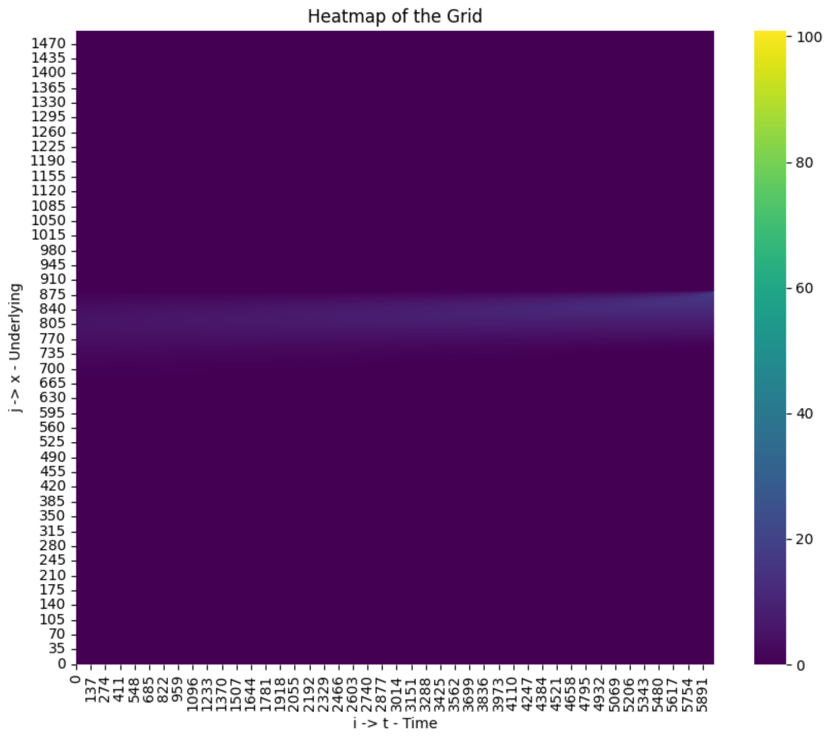


Figure 2: European Up-and-out Barrier Call Option (Table 1) Explicit Finite Differences Mesh Heatmap

(b) Boundary Conditions

The terminal condition is implemented as stated in the definition of the problem. Whereas in the vanilla call we initialise the grid at maturity, for this barrier option we must additionally apply the barrier condition at maturity, as well as at each time step as we propagate backwards through the grid. The boundary conditions (in x) remain unchanged.
The full set of conditions for the barrier option are defined explicitly below (8).

(c) Code Development

First, we implement the general form of a parabolic partial differential equation:

$$\frac{\partial v(t,x)}{\partial t} = a(t,x)\frac{\partial^2 v(t,x)}{\partial x^2} + b(t,x)\frac{\partial v(t,x)}{\partial x} + c(x,t)v(t,x) + d(t,x) \tag{5}$$

for $(t,x) \in [0,T] \times [x_l, x_u]$. With the following boundary conditions:

$$\begin{aligned} v(T,x) &= f(x), \\ v(t,x_l) &= f_l(t), \\ v(t,x_u) &= f_u(t) \end{aligned}$$

where $f : [x_l, x_u] \to \mathbb{R}$ is the terminal boundary condition and $f_l, f_u : [0,T] \to \mathbb{R}$ are lower and upper boundary conditions respectively.

The Black-Scholes equation is a special case of (5) where $v(t, S(t)) : [0,T] \times \mathbb{R} \to \mathbb{R}$ is a $C^{1,2}$ pricing function for an underlying asset with price $S(t)$ at time t, with parameters:

$$\begin{aligned} a(t,z) &= -\frac{\sigma(t,z)^2}{2}z^2, \\ b(t,z) &= -r_f z, \\ c(t,z) &= r_f, \\ d(t,z) &= 0. \end{aligned}$$

with $\sigma(t,z)$ given by (2).

The boundary conditions for a call option with expiry date T, strike price K and barrier D are

$$\begin{aligned} f(z) = v(T,z) &= \begin{cases} \max(z-K,0), & \text{if } \max\limits_{0\le t\le T} S_t < D, \\ 0, & \text{otherwise.} \end{cases} \\ f_u(t) = v(t,z_u) &= z_u - e^{-r_f(T-t)K} \\ f_l(t) = v(t,z_l) &= 0 \end{aligned}$$

We solve this parabolic PDE (Black-Scholes equation) using an explicit finite differences scheme. First defining a finite set $\{(t_i, x_j) : i \in [0, i_{max}], j \in [0, j_{max}]\}$ and taking $t_i = i\Delta t$, $x_j = x_l + j\Delta x$, where $\Delta t = \frac{T}{i_{max}}$ and $\Delta x = \frac{x_u - x_l}{j_{max}}$. The method operates on the following recursive formula:

$$v_{i-1,j} = A_{i,j}v_{i,j-1} + B_{i,j}v_{i,j} + C_{i,j}v_{i,j+1} + D_{i,j}$$

where we are using the notation $X_{i,j} = X(t_i, x_j)$ such that we have $f_i = f(x_j)$, $f_{l,i} = f_l(t_i)$, $f_{u,i} = f_u(t_i)$ as conditions, and

$$A_{i,j} = \frac{\Delta t}{\Delta x}\left(\frac{b_{i,j}}{2} - \frac{a_{i,j}}{\Delta x}\right), \qquad B_{i,j} = 1 - \Delta t c_{i,j} + \frac{2\Delta t a_{i,j}}{\Delta x^2}$$
$$C_{i,j} = -\frac{\Delta t}{\Delta x}\left(\frac{b_{i,j}}{2} + \frac{a_{i,j}}{\Delta x}\right), \qquad D_{i,j} = -\Delta t d_{i,j}$$

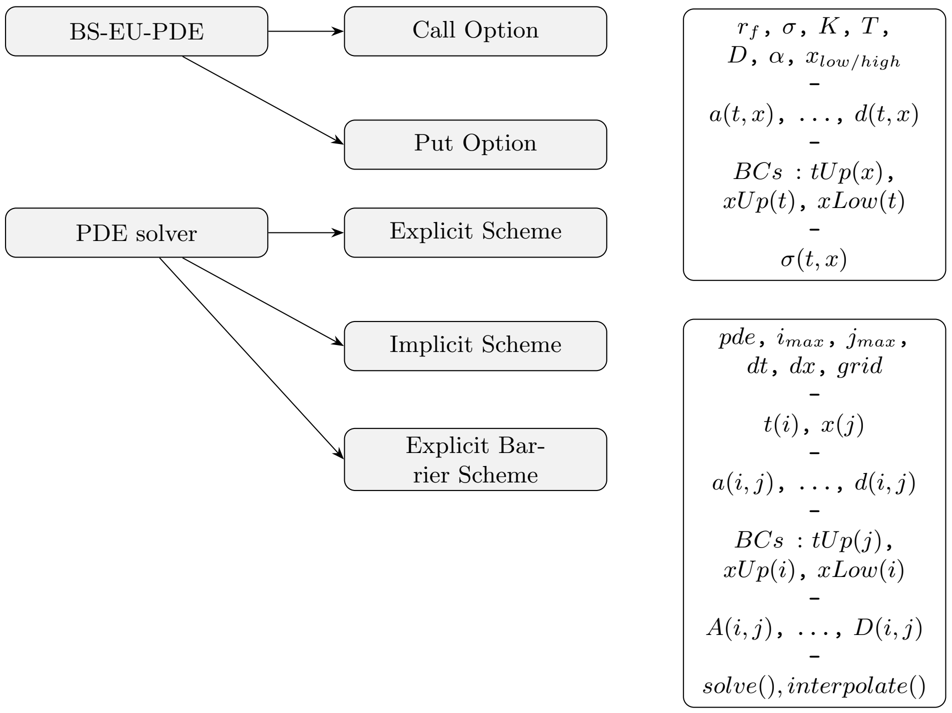
Since the following $v_{i,j}$ can be computed from the boundary conditions, we start from $i = i_{max}$ and finish with $i = 0$.

$$v_{i_{max},j} = f_i, \forall j \in [0, j_{max}]$$

Then using the boundary conditions we can also show that

$$v_{i-1,0} = f_{l,i-1}, \tag{6}$$
$$v_{i-1,j_{max}} = f_{u,i-1} \tag{7}$$
$$v_{i-1,j} = \begin{cases} v_{i,j} & \text{if } x_j > D, \\ 0 & \text{otherwise.} \end{cases}, \forall j \in [0, j_{max}] \tag{8}$$

The structure of the code is demonstrated below with a class graph. On the right you will observe the data belonging to each object in the first line, and then following on, the relevant methods of that class. The inheritance structure is demonstrated with unidirectional edges. Both BlackScholesEUPDE and PDESolver are abstract classes, providing much of the core functionality of their children. In addition to the methods listed below, there are also multiple visualisations included in PDE solver.


BS-EU-PDE
Call Option
Put Option
PDE solver
Explicit Scheme
Implicit Scheme
Explicit Barrier Scheme
r_f, σ, K, T, D, α, $x_{low/high}$ - $a(t,x)$, ..., $d(t,x)$ - BCs : $tUp(x)$, $xUp(t)$, $xLow(t)$ - $\sigma(t,x)$
pde, i_{max}, j_{max}, dt, dx, $grid$ - $t(i)$, $x(j)$ - $a(i,j)$, ..., $d(i,j)$ - BCs : $tUp(j)$, $xUp(i)$, $xLow(i)$ - $A(i,j)$, ..., $D(i,j)$ - $solve(), interpolate()$

(c) Conditional Stability

To demonstrate that the explicit finite difference scheme for the Black-Scholes Partial Differential Equation (PDE) is conditionally stable and to find the condition under which the scheme would be stable, we need to analyze the numerical stability of the explicit method applied to the PDE.

Explicit Finite Difference Scheme

The explicit finite difference scheme for the Black-Scholes PDE is given by:

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} + \frac{\sigma(t_n, S_i)^2 S_i^2}{2} \frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta S)^2} + rS_i \frac{V_{i+1}^n - V_{i-1}^n}{2\Delta S} - rV_i^n = 0$$

Rearranging, we get:

$$V_i^{n+1} = V_i^n + \Delta t \left[\frac{\sigma(t_n, S_i)^2 S_i^2}{2} \frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta S)^2} + rS_i \frac{V_{i+1}^n - V_{i-1}^n}{2\Delta S} - rV_i^n \right]$$

Stability Analysis

To analyze the stability, we can look at the maximum change in V as the time step progresses. This involves considering the coefficients in the above scheme. For stability, we typically require that the change in V at each time step is bounded.

The explicit scheme will be stable if the coefficients of the terms in V_{i+1}^n, V_i^n, and V_{i-1}^n are non-negative. To find the stability condition, we can rewrite the update formula as:

$$V_i^{n+1} = a_i V_{i-1}^n + b_i V_i^n + c_i V_{i+1}^n$$

where:

$$a_i = \Delta t \left[\frac{\sigma(t_n, S_i)^2 S_i^2}{2(\Delta S)^2} - \frac{rS_i}{2\Delta S} \right]$$

$$b_i = 1 - \Delta t \left[\frac{\sigma(t_n, S_i)^2 S_i^2}{(\Delta S)^2} + r \right]$$

$$c_i = \Delta t \left[\frac{\sigma(t_n, S_i)^2 S_i^2}{2(\Delta S)^2} + \frac{rS_i}{2\Delta S} \right]$$

For stability, we need $a_i \geq 0$, $b_i \geq 0$, $c_i \geq 0$, and $a_i + b_i + c_i \leq 1$. This implies:

$$\Delta t \leq \frac{2(\Delta S)^2}{\sigma(t_n, S_i)^2 S_i^2}$$

$$\Delta t \leq \frac{2}{r + \frac{\sigma(t_n, S_i)^2 S_i^2}{(\Delta S)^2}}$$

The most restrictive of these conditions typically determines the allowable time step Δt. Therefore, the stability condition is:

$$\Delta t \leq \min \left(\frac{2(\Delta S)^2}{\sigma(t_n, S_i)^2 S_i^2}, \frac{2}{r + \frac{\sigma(t_n, S_i)^2 S_i^2}{(\Delta S)^2}} \right)$$

Specific Volatility Surface

Given the local volatility surface:

$$\sigma(t, S_t) = 0.13e^{-\frac{t}{100}}$$

We substitute this into the stability condition. The first part of the condition becomes:

$$\Delta t \leq \frac{2(\Delta S)^2}{(0.13e^{-\frac{t_n}{100}})^2 S_i^2} = \frac{2(\Delta S)^2 e^{\frac{2t_n}{100}}}{0.0169 S_i^2}$$

Therefore, the condition for stability with the given volatility surface is:

$$\Delta t \leq \frac{2(\Delta S)^2 e^{\frac{2t_n}{100}}}{0.0169 S_i^2}$$

Thus, the explicit scheme is conditionally stable, and the stability condition depends on the time step Δt, the stock price step ΔS, and the specific form of the local volatility surface.

(d) Barrier Options vs Vanilla Options

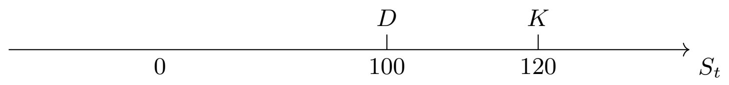
Option premiums are lower for prospective holders as the risk to the option writers is reduced. This cheaper price means that the contract has 'less' optionality than the vanilla counterpart; the exercise window is reduced.

The exotic nature of barrier options allows a holder greater customisation, typically in the payoff structure. This entails the drawback that the market is much less liquid compared to that of vanilla options.

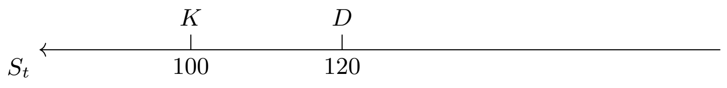
In general, the higher degree of sophistication lends itself to established actors, with more acute interests and greater modelling capacity/ experience.

(e) More barrier options

As in the name, an up-and-out call is out the money permanently if at any time $D \leq S_t$. If $D \leq K$ then intuitively the option has no value. A call option is in the money when $S_t > K$, this condition violates the barrier condition (the first inequality) and therefore we can never have a positive payoff.



The same is true of the reciprocal derivative, the down-and-out put. The payoff structure works the same but inverted in the x-axis (think of traditional vanilla call/ put payoff graphs). We can therefore never receive a positive payoff as the barrier condition is exercised prior to reaching the strike.



Question 3

(a) Heston PDE

(i)

The stochastic differential equations (SDEs) for the underlying stock price S and variance ν are given by:

$$\begin{cases} dS_t = \mu S_t dt + \sqrt{\nu_t} S_t dW_{1,t} \\ d\nu_t = \kappa(\theta - \nu_t)dt + \sigma\sqrt{\nu_t} dW_{2,t} \end{cases}$$

with $dW_{1,t} \cdot dW_{2,t} = \rho dt$.

The portfolio consists of:

$$\Pi = V(S,\nu,t) + \Delta S + \phi U$$

Using Ito's Lemma, the dynamics of $V(S,\nu,t)$ can be written as:

$$dV = \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}dS + \frac{\partial V}{\partial \nu}d\nu + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(dS)^2 + \frac{1}{2}\frac{\partial^2 V}{\partial \nu^2}(d\nu)^2 + \frac{\partial^2 V}{\partial S \partial \nu}dSd\nu$$

Similarly, for $U(S,\nu,t)$:

$$dU = \frac{\partial U}{\partial t}dt + \frac{\partial U}{\partial S}dS + \frac{\partial U}{\partial \nu}d\nu + \frac{1}{2}\frac{\partial^2 U}{\partial S^2}(dS)^2 + \frac{1}{2}\frac{\partial^2 U}{\partial \nu^2}(d\nu)^2 + \frac{\partial^2 U}{\partial S \partial \nu}dSd\nu$$

Substitute the given SDEs for dS and $d\nu$ into the expressions for dV and dU:

$$\begin{aligned} dV &= \left(\frac{\partial V}{\partial t} + \frac{\partial V}{\partial S}\mu S + \frac{\partial V}{\partial \nu}\kappa(\theta-\nu) + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\nu S^2 + \frac{1}{2}\frac{\partial^2 V}{\partial \nu^2}\sigma^2\nu + \frac{\partial^2 V}{\partial S\partial \nu}\rho\sigma\nu S\right)dt \\ &\quad + \left(\frac{\partial V}{\partial S}\sqrt{\nu}S\right)dW_1 + \left(\frac{\partial V}{\partial \nu}\sigma\sqrt{\nu}\right)dW_2 \\ dU &= \left(\frac{\partial U}{\partial t} + \frac{\partial U}{\partial S}\mu S + \frac{\partial U}{\partial \nu}\kappa(\theta-\nu) + \frac{1}{2}\frac{\partial^2 U}{\partial S^2}\nu S^2 + \frac{1}{2}\frac{\partial^2 U}{\partial \nu^2}\sigma^2\nu + \frac{\partial^2 U}{\partial S\partial \nu}\rho\sigma\nu S\right)dt \\ &\quad + \left(\frac{\partial U}{\partial S}\sqrt{\nu}S\right)dW_1 + \left(\frac{\partial U}{\partial \nu}\sigma\sqrt{\nu}\right)dW_2 \end{aligned}$$

The portfolio value Π dynamics are:

$$d\Pi = dV + \Delta dS + \phi dU$$

Substituting dS and dU in the portfolio gives:

$$\begin{aligned} d\Pi = \Big[&\frac{\partial V}{\partial t} + \frac{\partial V}{\partial S}\mu S + \frac{\partial V}{\partial \nu}\kappa(\theta-\nu) + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\nu S^2 + \frac{1}{2}\frac{\partial^2 V}{\partial \nu^2}\sigma^2\nu + \frac{\partial^2 V}{\partial S\partial \nu}\rho\sigma\nu S \\ &+ \Delta\mu S_t \\ &+ \phi\left(\frac{\partial U}{\partial t} + \frac{\partial U}{\partial S}\mu S + \frac{\partial U}{\partial \nu}\kappa(\theta-\nu) + \frac{1}{2}\frac{\partial^2 U}{\partial S^2}\nu S^2 + \frac{1}{2}\frac{\partial^2 U}{\partial \nu^2}\sigma^2\nu + \frac{\partial^2 U}{\partial S\partial \nu}\rho\sigma\nu S\right)\Big]dt \\ &+ \left(\frac{\partial V}{\partial S}\sqrt{\nu}S + \Delta\sqrt{\nu}S + \phi\frac{\partial U}{\partial S}\sqrt{\nu}S\right)dW_1 + \left(\frac{\partial V}{\partial \nu}\sigma\sqrt{\nu} + \phi\frac{\partial U}{\partial \nu}\sigma\sqrt{\nu}\right)dW_2 \end{aligned}$$

To eliminate the risk, we set the coefficients of dW_1 and dW_2 to zero:

$$\begin{cases} \frac{\partial V}{\partial S} + \Delta + \phi \frac{\partial U}{\partial S} = 0 \\ \frac{\partial V}{\partial \nu} + \phi \frac{\partial U}{\partial \nu} = 0 \end{cases}$$

Solving these gives:

$$\begin{cases} \Delta = \frac{\partial V}{\partial \nu} / \frac{\partial U}{\partial \nu} \frac{\partial U}{\partial S} - \frac{\partial V}{\partial S} \\ \phi = -\frac{\partial V}{\partial \nu} / \frac{\partial U}{\partial \nu} \end{cases}$$

Substitute Δ and ϕ back into the portfolio dynamics and equating equation $d\Pi = r\Pi dt$, we get the PDE:

$$\begin{aligned} rV + rS\left(\frac{V_\nu}{U_\nu}U_S - V_S\right) - r\frac{V_\nu}{U_\nu}U = {} & V_t + V_S\mu S + V_\nu\kappa(\theta - \nu) + \frac{1}{2}V_{SS}\nu S^2 + \frac{1}{2}V_{\nu\nu}\sigma^2\nu + V_{S\nu}\rho\sigma\nu S \\ & + \left(\frac{V_\nu}{U_\nu}U_S - V_S\right)\mu S \\ & - \frac{V_\nu}{U_\nu}\left(\frac{\partial U}{\partial t} + U_S\mu S + U_\nu\kappa(\theta - \nu) + \frac{1}{2}U_{SS}\nu S^2 + \frac{1}{2}U_{\nu\nu}\sigma^2\nu + V_{S\nu}\rho\sigma\nu S\right) \end{aligned}$$

$$\frac{\left[\frac{\partial V}{\partial t} + \frac{\nu S^2}{2}\frac{\partial^2 V}{\partial S^2} + \rho\sigma\nu S\frac{\partial^2 V}{\partial S\partial\nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 V}{\partial \nu^2} - rV + rS\frac{\partial V}{\partial S}\right]}{\frac{\partial V}{\partial \nu}} = \frac{\left[\frac{\partial U}{\partial t} + \frac{\nu S^2}{2}\frac{\partial^2 U}{\partial S^2} + \rho\sigma\nu S\frac{\partial^2 U}{\partial S\partial\nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 U}{\partial \nu^2} - rU + rS\frac{\partial U}{\partial S}\right]}{\frac{\partial U}{\partial \nu}}$$

(ii)

The left-hand side (LHS) is a function of V only, and the right-hand side (RHS) is a function of U only. The only way that this can be is for both sides to be written as some function f of independent variables S, ν and t. Without loss of generality, the choice of $f(S, \nu, t)$ is arbitrary. As in Heston (1993), we specify the function as:

$$f(S, \nu, t) = -\kappa(\theta - \nu) + \lambda(S, \nu, t),$$

where $\lambda(S, \nu, t)$ is the market price of volatility (variance) risk. Substitute the LHS with $f(S, \nu, t)$ to obtain the PDE:

$$\left[\frac{\partial V}{\partial t} + \frac{\nu S^2}{2}\frac{\partial^2 V}{\partial S^2} + \rho\sigma\nu S\frac{\partial^2 V}{\partial S\partial\nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 V}{\partial \nu^2} - rV + rS\frac{\partial V}{\partial S}\right] + [\kappa(\theta - \nu) - \lambda(S, \nu, t)]\frac{\partial V}{\partial \nu} = 0$$

(b) Heston Price

(i)

Here we define the log price $x = \log S$. The original PDE in terms of S:

$$\frac{\partial U}{\partial t} + \frac{\nu S^2}{2}\frac{\partial^2 U}{\partial S^2} + \rho\sigma\nu S\frac{\partial^2 U}{\partial S\partial \nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 U}{\partial \nu^2} - rU + rS\frac{\partial U}{\partial S} + [\kappa(\theta-\nu) - \lambda(S,\nu,t)]\frac{\partial U}{\partial \nu} = 0$$

$$Since : x = \log S \implies S = e^x$$

We get the following partial derivatives:

$$\frac{\partial U}{\partial S} = \frac{1}{S}\frac{\partial U}{\partial x}$$

$$\frac{\partial^2 U}{\partial S^2} = \frac{1}{S^2}\left(\frac{\partial U^2}{\partial x^2} - \frac{\partial U}{\partial x}\right)$$

Substitute transformed partial derivatives into the PDE:

$$\frac{\partial U}{\partial t}+\frac{\nu e^{2x}}{2}\cdot\frac{1}{e^{2x}}\left(\frac{\partial^2 U}{\partial x^2} - \frac{\partial U}{\partial x}\right)+\rho\sigma\nu e^x\cdot\frac{1}{e^x}\frac{\partial^2 U}{\partial x\partial \nu}+\frac{\sigma^2\nu}{2}\frac{\partial^2 U}{\partial \nu^2}-rU+re^x\cdot\frac{1}{e^x}\frac{\partial U}{\partial x}+[\kappa(\theta-\nu) - \lambda(e^x,\nu,t)]\frac{\partial U}{\partial \nu} = 0$$

Simplify the PDE:

$$\frac{\partial U}{\partial t} + \frac{\nu}{2}\left(\frac{\partial^2 U}{\partial x^2} - \frac{\partial U}{\partial x}\right) + \rho\sigma\nu\frac{\partial^2 U}{\partial x\partial \nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 U}{\partial \nu^2} - rU + r\frac{\partial U}{\partial x} + [\kappa(\theta-\nu) - \lambda(e^x,\nu,t)]\frac{\partial U}{\partial \nu} = 0$$

After combining and rearranging terms we finally have the PDE:

$$\frac{\partial U}{\partial t} + \frac{\nu}{2}\frac{\partial^2 U}{\partial x^2} + \left(r - \frac{\nu}{2}\right)\frac{\partial U}{\partial x} + \rho\sigma\nu\frac{\partial^2 U}{\partial x\partial \nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 U}{\partial \nu^2} - rU + [\kappa(\theta-\nu) - \lambda(e^x,\nu,t)]\frac{\partial U}{\partial \nu} = 0$$

(ii)

We assume the market price of volatility (variance) risk is a linear function of variance $\lambda(S,\nu,t) = \lambda\nu$. Using $x = x_t = \log S$, $C_t = e^x P_1 - Ke^{-rT}P_2$, we have the PDE:

$$\frac{\partial C}{\partial t} + \frac{\nu}{2}\frac{\partial^2 C}{\partial x^2} + \left(r - \frac{\nu}{2}\right)\frac{\partial C}{\partial x} + \rho\sigma\nu\frac{\partial^2 C}{\partial x\partial \nu} + \frac{\sigma^2\nu}{2}\frac{\partial^2 C}{\partial \nu^2} - rC + [\kappa(\theta-\nu) - \lambda\nu]\frac{\partial C}{\partial \nu} = 0$$

(1) Deriving partial derivatives

$$\frac{\partial C}{\partial t} = e^x\frac{\partial P_1}{\partial t} - \left(Kre^{-rt}P_2 + Ke^{-rt}\frac{\partial P_2}{\partial t}\right)$$

$$\frac{\partial C}{\partial x} = e^x P_1 + e^x\frac{\partial P_1}{\partial x} - Ke^{-rt}\frac{\partial P_2}{\partial x}$$

$$\frac{\partial C}{\partial v} = e^x\frac{\partial P_1}{\partial v} - Ke^{-rt}\frac{\partial P_2}{\partial v}$$

$$\frac{\partial^2 C}{\partial x^2} = e^x P_1 + 2e^x\frac{\partial P_1}{\partial x} + e^x\frac{\partial^2 P_1}{\partial x^2} - Ke^{-rt}\frac{\partial^2 P_2}{\partial x^2}$$

$$\frac{\partial^2 C}{\partial v^2} = e^x \frac{\partial^2 P_1}{\partial v^2} - Ke^{-rt}\frac{\partial^2 P_2}{\partial v^2}$$

$$\frac{\partial^2 C}{\partial x \partial v} = e^x \frac{\partial P_1}{\partial v} + e^x \frac{\partial^2 P_1}{\partial v \partial x} - Ke^{-rt}\frac{\partial^2 P_2}{\partial v \partial x}$$

(2) Substituting these terms into the PDE, we get the two new PDEs for P_1 and P_2:

For P_1:

$$\frac{\partial P_1}{\partial t} + \rho\nu\frac{\partial^2 P_1}{\partial x \partial \nu} + \frac{\nu}{2}\frac{\partial^2 P_1}{\partial x^2} + \frac{\sigma^2\nu}{2}\frac{\partial^2 P_1}{\partial \nu^2} + \left(r + \frac{1}{2}\nu\right)\frac{\partial P_1}{\partial x} + (\kappa\theta - (\kappa + \lambda - \rho\sigma)\nu)\frac{\partial P_1}{\partial \nu} = 0$$

For P_2:

$$\frac{\partial P_2}{\partial t} + \rho\nu\frac{\partial^2 P_2}{\partial x \partial \nu} + \frac{\nu}{2}\frac{\partial^2 P_2}{\partial x^2} + \frac{\sigma^2\nu}{2}\frac{\partial^2 P_2}{\partial \nu^2} + \left(r - \frac{1}{2}\nu\right)\frac{\partial P_2}{\partial x} + (\kappa\theta - (\kappa + \lambda)\nu)\frac{\partial P_2}{\partial \nu} = 0$$

For notational convenience, these two PDEs can be combined into a single equation:

$$\frac{\partial P_j}{\partial t} + \rho\nu\frac{\partial^2 P_j}{\partial x \partial \nu} + \frac{\nu}{2}\frac{\partial^2 P_j}{\partial x^2} + \frac{\sigma^2\nu}{2}\frac{\partial^2 P_j}{\partial \nu^2} + (r + u_j\nu)\frac{\partial P_j}{\partial x} + (a - b_j\nu)\frac{\partial P_j}{\partial \nu} = 0,$$

for $j = 1, 2$ where:

$$u_1 = \frac{1}{2}, \quad u_2 = -\frac{1}{2}, \quad a = \kappa\theta, \quad b_1 = \kappa + \lambda - \rho\sigma, \quad \text{and} \quad b_2 = \kappa + \lambda.$$

(iii)

Given the characteristic function proposed in Heston (1993):

$$f_j(x, \nu, t; \phi) = \exp(C_j(\tau, \phi) + D_j(\tau, \phi)\nu_t + i\phi x_t)$$

(1) Deriving partial derivatives

$$\frac{\partial f_j}{\partial \tau} = \left(\frac{\partial C_j}{\partial \tau} + \nu_t\frac{\partial D_j}{\partial \tau}\right)\exp(C_j + D_j\nu_t + i\phi x_t) = \left(\frac{\partial C_j}{\partial \tau} + \nu_t\frac{\partial D_j}{\partial \tau}\right) f_j$$

$$\frac{\partial f_j}{\partial x} = (i\phi)\exp(C_j + D_j\nu_t + i\phi x_t) = i\phi f_j$$

$$\frac{\partial f_j}{\partial \nu} = D_j \exp(C_j + D_j\nu_t + i\phi x_t) = D_j f_j$$

$$\frac{\partial^2 f_j}{\partial x^2} = (i\phi)^2 \exp(C_j + D_j\nu_t + i\phi x_t) = -\phi^2 f_j$$

$$\frac{\partial^2 f_j}{\partial \nu^2} = D_j^2 \exp(C_j + D_j\nu_t + i\phi x_t) = D_j^2 f_j$$

$$\frac{\partial^2 f_j}{\partial x \partial \nu} = (i\phi D_j)\exp(C_j + D_j\nu_t + i\phi x_t) = i\phi D_j f_j$$

(2) Substituting the expressions for the derivatives into the PDE:

$$-\frac{\partial f_j}{\partial \tau} + \rho\sigma\nu \frac{\partial^2 f_j}{\partial x \partial \nu} + \frac{\nu}{2}\frac{\partial^2 f_j}{\partial x^2} + \frac{\sigma^2 \nu}{2}\frac{\partial^2 f_j}{\partial \nu^2} + (r + u_j \nu)\frac{\partial f_j}{\partial x} + (a - b_j \nu)\frac{\partial f_j}{\partial \nu} = 0$$

$$-\left(\frac{\partial C_j}{\partial \tau} + \nu \frac{\partial D_j}{\partial \tau}\right) f_j + \rho\sigma\nu(i\phi D_j f_j) + \frac{\nu}{2}(-\phi^2 f_j) + \frac{\sigma^2 \nu}{2}(D_j^2 f_j) + (r + u_j \nu)(i\phi f_j) + (a - b_j \nu)(D_j f_j) = 0$$

Dividing through by f_j:

$$-\left(\frac{\partial C_j}{\partial \tau} + \nu \frac{\partial D_j}{\partial \tau}\right) + \rho\sigma\nu(i\phi D_j) + \frac{\nu}{2}(-\phi^2) + \frac{\sigma^2 \nu}{2}(D_j^2) + (r + u_j \nu)(i\phi) + (a - b_j \nu)(D_j) = 0$$

Grouping terms by powers of ν:

$$-\frac{\partial C_j}{\partial \tau} - \nu \frac{\partial D_j}{\partial \tau} + \rho\sigma i\phi\nu D_j - \frac{\nu}{2}\phi^2 + \frac{\sigma^2 \nu}{2} D_j^2 + ri\phi + u_j \nu i\phi + aD_j - b_j \nu D_j = 0$$

Separating terms without ν:

$$-\frac{\partial C_j}{\partial \tau} + ri\phi + aD_j = 0$$

And terms with ν:

$$-\nu \frac{\partial D_j}{\partial \tau} + \rho\sigma i\phi\nu D_j - \frac{\nu}{2}\phi^2 + \frac{\sigma^2 \nu}{2} D_j^2 + u_j \nu i\phi - b_j \nu D_j = 0$$

Simplifying, we get two equations:
For C_j:

$$\frac{\partial C_j}{\partial \tau} = ri\phi + aD_j$$

For D_j:

$$\frac{\partial D_j}{\partial \tau} = \rho\sigma i\phi D_j - \frac{\phi^2}{2} + \frac{\sigma^2}{2} D_j^2 + u_j i\phi - b_j D_j$$

These are the two differential equations for C_j and D_j in the Heston model.

Question 4

(a) Volatility Smile

Volatility smile occurs when OTM options exhibit higher implied volatilities compared to ATM options. In other words, options with strike prices further from the current market price show increased implied volatilities.

Fear and Greed Factor are the main reason of that. People react more violently to big events or news. During periods of heightened uncertainty or significant negative market events or market distress, investors often rush to buy OTM put options as a hedge against potential losses. This increased demand for downside protection elevates the implied volatility of these options. Conversely, during periods of optimism and bullish sentiment, the implied volatility of OTM call options can rise, so finally this phenomenon produces a similar pattern of smiles.

In equity markets, investors typically view downside risk as greater than upside potential. This is because stock prices can only fall to zero, but they can theoretically rise indefinitely. Consequently, investors are often willing to pay more for put options, which gain value as prices drop, than for call options, which gain value as prices rise. This preference can lead to higher implied volatility for OTM put options, creating a volatility skew.

(b) Options on Volatility

Deep OTM options have a low value. When volatility decreases, the price of deep OTM options also decreases, but only slightly, as their value can never go below zero. However, when volatility increases, the price of deep OTM options increases significantly. From this perspective, deep OTM options have a similar graph to a volatility option, resembling a convex shape.

(c) Heston Model Fit

The Heston model addresses the limitations of the Black-Scholes model by introducing stochastic volatility, making it more suitable for real financial markets rather than assuming constant volatility. It has limitations for short-term maturity options for several reasons:

1) Heston model assume that the volatility is a mean-reverting process, which means it tends to revert to a long-term average over time. In the short term, this mean-reversion feature may not be able to account for sudden spikes or drops in volatility that can occur in the market.

2) Heston model is calibrated to fit market data, but it can be challenging for short-term options because the time to expiry is too short and market conditions can change rapidly, so it struggles to predict the prices of short-term options as the model fails to capture the high implied volatility.

1 Appendix

1.1 Figures

S_0	K	r_f	D	T (years)	σ	P
\$100	\$100	5.0%	\$118	0.083	0.20	\$2.60

Table 2: Vanilla European Call, Parameters and Price

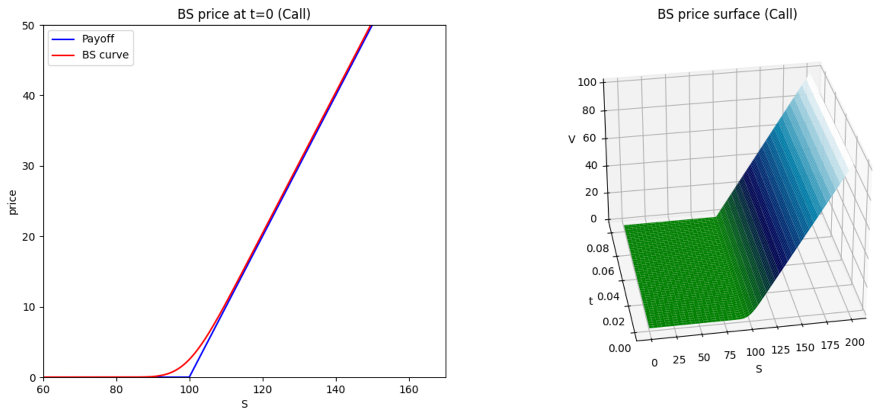


Figure 3: Vanilla European Call Option (Table 1) Payoff and Price Vs. Underlying (left), V(t, S) - Price Surface, t - Time, S - Underlying (right)

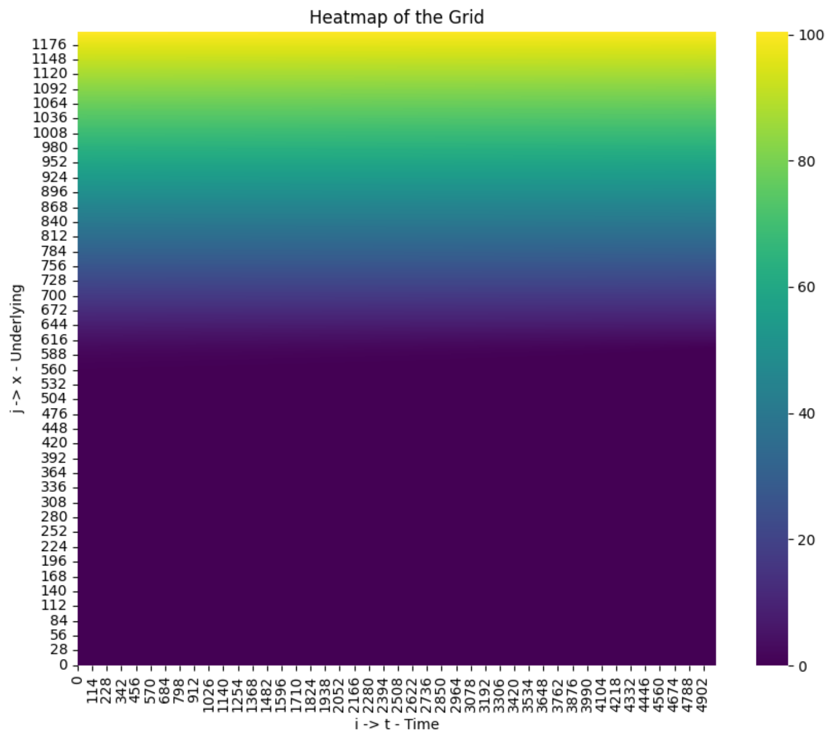


Figure 4: Vanilla European Call Option (Table 1) Explicit Finite Differences Mesh Heatmap

1.2 Code

```
import numpy as np
import scipy.sparse as sparse
import math
import matplotlib.pyplot as plt
from matplotlib import cm
from seaborn import heatmap

class EuBlackScholesOption:
    def __init__(self, rate, sigma, strike, t_up, x_low, x_up, barrier):
        self.rate = rate
        self.sigma = sigma
        self.strike = strike
        self.t_up = t_up
        self.x_low = x_low
        self.x_up = x_up
        self.D = barrier

    def sigma_calc(self,t,x):
        if self.D is not None:
            return 0.13 * math.exp(-t) * (100 / x) ** self.alpha
        return self.sigma

    def a(self, t, x):
        if self.D is not None:
            return -self.sigma_calc(t,x)**2*x**2/2
        return -self.sigma**2*x**2/2

    def b(self, t, x):
        return -self.rate*x

    def c(self, t, x):
        return self.rate

    def d(self, t, x):
        return 0

class EuPutBlackScholes(EuBlackScholesOption):
    def __init__(self, *, rate, sigma, strike, t_up, x_low, x_up, barrier=None):
        super().__init__(rate, sigma, strike, t_up, x_low, x_up, barrier)
        self.type = 'put'

    def bc_tup(self, x):
        return max(self.strike - x, 0)

    def bc_xup(self, t):
        return 0

    def bc_xlow(self, t):
        return math.exp(-self.rate * (self.t_up - t)) * self.strike

class EuCallBlackScholes(EuBlackScholesOption):
    def __init__(self, *, rate, sigma=None, strike, t_up, x_low, x_up, barrier=None, alpha=0.24):
        super().__init__(rate, sigma, strike, t_up, x_low, x_up, barrier)
        self.type = 'call'
        self.alpha = alpha

    def bc_tup(self, x):
        return max(x - self.strike, 0)

    def bc_xup(self, t):
        return self.x_up - self.strike * math.exp(-self.rate * (self.t_up - t))

    def bc_xlow(self, t):
        return 0

class PDESolver:
    """ Base class for all parabolic PDE solvers """
    def __init__(self, pde, imax, jmax):
        self.pde = pde
        self.imax = imax
        self.jmax = jmax
        self.dt = pde.t_up / imax
        self.dx = (pde.x_up - pde.x_low) / jmax
        self.grid = np.empty((imax + 1, jmax + 1))

    def t(self, i):
        return i * self.dt

    def x(self, j):
        return j * self.dx + self.pde.x_low

    def a(self, i, j):
        return self.pde.a(self.t(i), self.x(j))

    def b(self, i, j):
        return self.pde.b(self.t(i), self.x(j))

    def c(self, i, j):
        return self.pde.c(self.t(i), self.x(j))

    def d(self, i, j):
        return self.pde.d(self.t(i), self.x(j))
```

```
def tup(self, j):
    return self.pde.bc_tup(self.x(j))

def xlow(self, i):
    return self.pde.bc_xlow(self.t(i))

def xup(self, i):
    return self.pde.bc_xup(self.t(i))

def interpolate(self, t, x):
    i = int(t / self.dt)
    j = int((x - self.pde.x_low) / self.dx)
    l0 = (t - self.dt * i) / self.dt
    l1 = 1 - l0
    w0 = (x - self.dx * j) / self.dx
    w1 = 1 - w0
    return (self.grid[i,j] * l0 * w0
            + self.grid[i+1,j] * l1 * w0
            + self.grid[i,j+1] * l0 * w1
            + self.grid[i+1,j+1] * l1 * w1)

def plot_results(self, rx=30, ry=-100, save_path=None):
    S_grid = np.linspace(self.pde.x_low, self.pde.x_up, self.jmax + 1)
    T_grid = np.linspace(0, self.pde.t_up, self.imax + 1)

    V = self.grid

    fig = plt.figure(figsize=(15, 6))

    # Plot Payoff and BS curve
    ax1 = fig.add_subplot(121)
    if self.pde.type == 'call':
        if self.pde.D is not None:
            payoff = [self.pde.bc_tup(s) if self.pde.bc_tup(s) < (self.pde.D - self.pde.strike) else 0 for s in S_grid]
        else:
            payoff = [self.pde.bc_tup(s) for s in S_grid]
    else:
        payoff = [self.pde.bc_tup(s) for s in S_grid]
    ax1.plot(S_grid, payoff, color="blue", label="Payoff")
    ax1.plot(S_grid, V[0, :], color="red", label="BS curve")
    ax1.set_xlim(60, 170)
    ax1.set_ylim(0, 50)
    ax1.set_xlabel("S")
    ax1.set_ylabel("price")
    ax1.legend(loc="upper left")
    ax1.set_title(f"BS price at t=0 ({self.pde.type.capitalize()})")

    # Plot the BS price surface
    X, Y = np.meshgrid(T_grid, S_grid)
    ax2 = fig.add_subplot(122, projection="3d")
    ax2.plot_surface(Y, X, V.T, cmap=cm.ocean)
    ax2.set_title(f"BS price surface ({self.pde.type.capitalize()})")
    ax2.set_xlabel("S")
    ax2.set_ylabel("t")
    ax2.set_zlabel("V")
    ax2.view_init(rx, ry)  # this function rotates the 3d plot

    if save_path:
        plt.savefig(save_path)

    plt.show()

def plot_heatmap(self, save_path=None):
    plt.figure(figsize=(10, 8))
    grid_to_plot = np.nan_to_num(self.grid).T  # Replace NaN with zero
    ax = heatmap(grid_to_plot, cmap='viridis', cbar=True)
    ax.invert_yaxis()  # Flip the y-axis
    ax.set_xlabel('i -> t - Time')
```

```
        ax.set_ylabel('j -> x - Underlying')
        plt.title("Heatmap of the Grid")
        if save_path:
            plt.savefig(save_path)

        plt.show()


class ExplicitScheme(PDESolver):
    """ Explicit scheme to solve parabolic PDEs """
    def __init__(self, pde, imax, jmax):
        super().__init__(pde, imax, jmax)
        self.solve_grid()

    def A(self, i, j):
        return self.dt/self.dx*(self.b(i,j)/2 - self.a(i,j)/self.dx)

    def B(self, i, j):
        return 1 - self.dt*self.c(i,j) + 2 * self.dt * self.a(i,j) / self.dx**2

    def C(self, i, j):
        return -self.dt / self.dx * (self.b(i,j) / 2 + self.a(i,j) / self.dx)

    def D(self, i, j):
        return -self.dt * self.d(i,j)

    def solve_grid(self):
        def grid(i, j):
            return self.A(i,j) * self.grid[i,j-1] + self.B(i,j) * self.grid[i,j] + self.C(i,j) * self.grid[i,j+1] + self.D(i,j)

        self.grid[self.imax, :] = [self.tup(j) for j in range(self.jmax+1)]
        for i in range(self.imax, 0, -1):
            self.grid[i-1, 0] = self.xlow(i-1)
            self.grid[i, self.jmax] = self.xup(i-1)
            self.grid[i-1, 1:-1] = [grid(i,j) for j in range(1, self.jmax)]

class ImplicitScheme(PDESolver):
    """ Explicit scheme to solve parabolic PDEs """
    def __init__(self, pde, imax, jmax):
        super().__init__(pde, imax, jmax)
        self.solve_grid()

    def A(self, i, j):
        return 0

    def B(self, i, j):
        return 1

    def C(self, i, j):
        return 0

    def D(self, i, j):
        return -self.dt * self.d(i-1, j)

    def E(self, i, j):
        return -self.dt / self.dx * (self.b(i-1, j) / 2 - self.a(i - 1, j) / self.dx)

    def F(self, i, j):
        return 1 + self.dt * self.c(i-1, j) - 2 * self.dt * self.a(i-1, j) / self.dx**2

    def G(self, i, j):
        return self.dt / self.dx * (self.b(i-1, j) / 2 + self.a(i-1, j) / self.dx)

    def w(self, i):
        return [self.D(i,1) + self.A(i,1) * self.xlow(i) - self.E(i,1) * self.xlow(i-1)] \
            + [self.D(i,k) for k in range(2, self.jmax - 1)] + [self.D(i,self.jmax-1) \
            + self.C(i, self.jmax-1) * self.xup(i) - self.G(i, self.jmax-1) * self.xup(i-1)]

    def compute_vi(self, i):
        A1 = [self.A(i,j) for j in range(2,self.jmax)]
        Ad = [self.B(i,j) for j in range(1, self.jmax)]
        Ar = [self.C(i,j) for j in range(1, self.jmax-1)]
        A = sparse.diags([A1, Ad, Ar], [-1, 0, 1])
        B1 = [self.E(i,j) for j in range(2, self.jmax)]
        Bd = [self.F(i,j) for j in range(1,self.jmax)]
        Br = [self.G(i,j) for j in range(1, self.jmax-1)]
        B = sparse.diags([B1, Bd, Br], [-1, 0, 1])
        rhs = A @ self.grid[i,1:-1] + self.w(i)
        return sparse.linalg.splu(B).solve(rhs)

    def solve_grid(self):
        self.grid[self.imax, :] = [self.tup(j) for j in range(self.jmax+1)]
        for i in range(self.imax, -1, -1):
            self.grid[i-1, 0] = self.xlow(i-1)
            self.grid[i, self.jmax] = self.xup(i-1)
            self.grid[i-1, 1:-1] = self.compute_vi(i)

class ExplicitBarrierScheme(PDESolver):
    """ Explicit scheme to solve parabolic PDEs with barrier options """
    def __init__(self, pde, imax, jmax):
        super().__init__(pde, imax, jmax)
        self.solve_grid()

    def A(self, i, j):
        return self.dt/self.dx * (self.b(i, j) / 2 - self.a(i, j) / self.dx)

    def B(self, i, j):
        return 1 - self.dt * self.c(i, j) + 2 * self.dt * self.a(i, j) / self.dx**2

    def C(self, i, j):
        return -self.dt / self.dx * (self.b(i, j) / 2 + self.a(i, j) / self.dx)

    def D(self, i, j):
        return -self.dt * self.d(i, j)

    def solve_grid(self):
        def grid(i, j):
            return (self.A(i, j) * self.grid[i, j-1] +
                    self.B(i, j) * self.grid[i, j] +
                    self.C(i, j) * self.grid[i, j+1] +
                    self.D(i, j))

        # Initialize grid at maturity
        self.grid[self.imax, :] = [self.tup(j) for j in range(self.jmax+1)]

        # Apply the barrier condition at maturity
        for j in range(self.jmax+1):
            if self.x(j) >= self.pde.D:
                self.grid[self.imax, j] = 0

        # Iterate backward in time
        for i in range(self.imax, 0, -1):
            self.grid[i-1, 0] = self.xlow(i-1)
            self.grid[i-1, -1] = self.xup(i-1)
            for j in range(1, self.jmax):
                self.grid[i-1, j] = grid(i, j)
                # Apply the barrier condition at each time step
                if self.x(j) >= self.pde.D:
                    self.grid[i-1, j] = 0

# Put-Call Parity Test
rate=0.05
sigma=0.2
strike=100
t_up=1/12
x_low=0
x_up=200
imax = 5000
```

```
jmax = 1200
S = 100  # Current stock price

# Create instances of put and call solvers
put_pde = EuPutBlackScholes(rate=rate, sigma=sigma, strike=strike, t_up=t_up, x_low=x_low, x_up=x_up)
call_pde = EuCallBlackScholes(rate=rate, sigma=sigma, strike=strike, t_up=t_up, x_low=x_low, x_up=x_up)

put_solver = ExplicitScheme(put_pde, imax, jmax)
call_solver = ExplicitScheme(call_pde, imax, jmax)

# Interpolate to find the option prices at S
put_price = put_solver.interpolate(0, S)
call_price = call_solver.interpolate(0, S)

# Calculate the expected put-call parity value
expected_value = S - strike * math.exp(-rate * t_up)

# Print the results
print(f"Put Price: {put_price:.2f}")
print(f"Call Price: {call_price:.2f}")
print(f"Put-Call Parity Difference: {(call_price - put_price - expected_value):.4f}")

call_solver.plot_results(save_path='figures/vanilla_call.png')
call_solver.plot_heatmap(save_path='figures/heatmap__vanilla_call.png')

implicit_put_solver = ImplicitScheme(put_pde, imax, jmax)
implicit_call_solver = ImplicitScheme(call_pde, imax, jmax)

# Interpolate to find the option prices at S
implicit_put_price = implicit_put_solver.interpolate(0, S)
implicit_call_price = implicit_call_solver.interpolate(0, S)

# Calculate the expected put-call parity value
expected_value = S - strike * math.exp(-rate * t_up)

# Print the results
print(f"Put Price: {put_price:.2f}")
print(f"Call Price: {call_price:.2f}")
print(f"Put-Call Parity Difference: {(implicit_call_price - implicit_put_price - expected_value):.4f}")

rate=0.018
# sigma=0.24
strike=100
t_up=1/2
x_low=0
x_up=200
barrier=118
imax = 6000
jmax = 1500
S = 100  # Current stock price

# Create instances of put and call solvers
barrier_pde = EuCallBlackScholes(rate=rate, strike=strike, t_up=t_up, x_low=x_low, x_up=x_up, barrier=barrier)

solver2 = ExplicitBarrierScheme(barrier_pde, imax, jmax)

# Interpolate to find the option prices at S
price = solver2.interpolate(0, S)

# Print the results
print(f"Barrier Call Price: {price:.3f}")

solver2.plot_heatmap(save_path='figures/heatmap_barrier_call.png')
```