Math 6635 Homework #2

Vidit Pokharna

March 11, 2025

1. An American put option gives the holder the right to sell the underlying asset at the strike price E at any time before or at expiration. If the holder exercises the put immediately, they receive $E - S$ if $E > S$, or zero if $S \geq E$, as it won't be exercised if $S \geq E$. Since the put option allows early exercise, its value must always be at least as much as the intrinsic value $\max(E - S, 0)$. Thus, the inequality:

$$P \geq \max(E - S, 0)$$

Consider a portfolio where we sell one call option and hold one share of stock. The value of this portfolio is $S - C$. If we had instead invested an amount $Ee^{-r(T-t)}$ in a risk-free bond, it would grow to E at expiration. At expiry, the call option is either exercised or not:

- If exercised, deliver the stock and receive E, leaving us with $E - C$
- If not exercised, just hold $S - C$

Ensuring that the present value of the strike price is accounted for AND since arbitrage opportunities should not exist, the call option price must satisfy:

$$C \geq S - Ee^{-r(T-t)}$$

Consider the portfolio $C - P + E - S$. If the put is exercised early, the payoff is $E - S$, while the call is worthless. At expiration, if $S > E$, the call is exercised and the put is worthless, leading to a final value of $S - E$. Since arbitrage opportunities should not exist, we get:

$$C - P \geq S - E$$

This comes from the put-call parity for European options:

$$C_E - P_E = S - Ee^{-r(T-t)}$$

Since an American option can never be worth less than a European one, we get:

$$C - P \leq S - Ee^{-r(T-t)}$$

Thus, using transitivity of inequalities, giving us:

$$S - E \leq C - P \leq S - Ee^{-r(T-t)}$$

When you exercise a call option early, you give up the time value of the option. Instead of exercising early, you can sell the call option in the market to capture its full value.

If the option is exercised early:

- You pay E and receive the stock worth S
- However, if you wait, you can earn risk-free interest on E while still retaining the option

This means the option is more valuable alive than exercised early. Therefore, in the absence of dividends, an American call option should never be exercised before expiration.

2.

$$\begin{aligned}
u(x_{i+1}) &= u(x_i) + \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) + \frac{\Delta x^3}{6}u'''(x_i) + O(\Delta x^4)\\
u(x_{i+1}) - u(x_i) &= \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) + O(\Delta x^3)\\
\frac{u(x_{i+1}) - u(x_i)}{\Delta x} &= u'(x_i) + \frac{\Delta x}{2}u''(x_i) + O(\Delta x^2)\\
\frac{u_{i+1} - u_i}{\Delta x} &= u'(x_i) + O(\Delta x)
\end{aligned}$$

$$\begin{aligned}
u(x_{i+1}) &= u(x_i) + \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) + \frac{\Delta x^3}{6}u'''(x_i) + O(\Delta x^4)\\
u(x_{i-1}) &= u(x_i) - \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) - \frac{\Delta x^3}{6}u'''(x_i) + O(\Delta x^4)\\
u(x_{i+1}) - u(x_{i-1}) &= \left(u(x_i) + \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) + \frac{\Delta x^3}{6}u'''(x_i) + O(\Delta x^4)\right)\\
&\quad - \left(u(x_i) - \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) - \frac{\Delta x^3}{6}u'''(x_i) + O(\Delta x^4)\right)\\
u(x_{i+1}) - u(x_{i-1}) &= 2\Delta x u'(x_i) + \frac{2\Delta x^3}{6}u'''(x_i) + O(\Delta x^4)\\
\frac{u(x_{i+1}) - u(x_{i-1})}{2\Delta x} &= u'(x_i) + \frac{\Delta x^2}{6}u'''(x_i) + O(\Delta x^4)\\
\frac{u_{i+1} - u_{i-1}}{2\Delta x} &= u'(x_i) + O(\Delta x^2)
\end{aligned}$$

$$\begin{aligned}
u(x_{i+1}) &= u(x_i) + \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) + \frac{\Delta x^3}{6}u'''(x_i) + \frac{\Delta x^4}{24}u''''(x_i) + O(\Delta x^5)\\
u(x_{i-1}) &= u(x_i) - \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) - \frac{\Delta x^3}{6}u'''(x_i) + \frac{\Delta x^4}{24}u''''(x_i) + O(\Delta x^5)\\
u(x_{i+1}) + u(x_{i-1}) &= \left(u(x_i) + \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) + \frac{\Delta x^3}{6}u'''(x_i) + \frac{\Delta x^4}{24}u''''(x_i)\right)\\
&\quad + \left(u(x_i) - \Delta x u'(x_i) + \frac{\Delta x^2}{2}u''(x_i) - \frac{\Delta x^3}{6}u'''(x_i) + \frac{\Delta x^4}{24}u''''(x_i)\right) + O(\Delta x^5)\\
u(x_{i+1}) + u(x_{i-1}) &= 2u(x_i) + \Delta x^2 u''(x_i) + \frac{\Delta x^4}{12}u''''(x_i) + O(\Delta x^5)\\
u''(x_i) &= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{\Delta x^2} - \frac{\Delta x^2}{12}u''''(x_i) + O(\Delta x^4)\\
\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} &= u''(x_i) + O(\Delta x^2)
\end{aligned}$$

3. (a)

$$\begin{aligned}
u_t &= \nu u_{xx}, \quad \nu > 0\\
\frac{U_j^{n+1}-U_j^n}{\Delta t} &= \nu \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}\\
\frac{U_j^{n+1}-U_j^n}{\Delta t} &= u_t + \frac{\Delta t}{2}u_{tt} + O(\Delta t^2)\\
\frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2} &= u_{xx} + \frac{\Delta x^2}{12}u_{xxxx} + O(\Delta x^4)\\
U_j^n &= G^n e^{ikj\Delta x}\\
\frac{G-1}{\Delta t} &= \nu \frac{e^{ik\Delta x} - 2 + e^{-ik\Delta x}}{\Delta x^2}\\
e^{ik\Delta x} - 2 + e^{-ik\Delta x} &= -4\sin^2(k\Delta x/2)\\
G &= 1 - 4\nu \frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)\\
\Delta t &\le \frac{\Delta x^2}{2\nu}
\end{aligned}$$

(b)

$$\begin{aligned}
\frac{U_j^{n+1}-U_j^n}{\Delta t} &= \frac{\nu}{2}\left(\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{\Delta x^2} + \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}\right)\\
U_j^n &= G^n e^{ikj\Delta x}\\
G &= \frac{1-2s\sin^2(k\Delta x/2)}{1+2s\sin^2(k\Delta x/2)}, \quad s = \frac{\nu\Delta t}{\Delta x^2}\\
|G| &\le 1
\end{aligned}$$

(c)

$$\begin{aligned}
\frac{U_j^{n+1}-U_j^n}{\Delta t} &= \frac{\nu}{2}\left(\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{\Delta x^2} + \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}\right)\\
&\quad + \frac{1}{2}\left\{\frac{U_{j+1}^{n+1} - U_{j-1}^{n+1}}{2\Delta x} + \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}\right\}\\
U_j^n &= G^n e^{ikj\Delta x}\\
|G| &\le 1
\end{aligned}$$

Scheme	Truncation Error	Stability
Explicit Scheme	$O(\Delta t, \Delta x^2)$	Conditionally stable, $\Delta t \le \frac{\Delta x^2}{2\nu}$
Crank-Nicolson (Heat Eq.)	$O(\Delta t^2, \Delta x^2)$	Unconditionally stable
Crank-Nicolson (Advective Heat Eq.)	$O(\Delta t^2, \Delta x^2)$	Unconditionally stable

4. The Black-Scholes equation in transformed time $\tau = T - t$ is given by:

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV$$
$$V_j^n \approx V(j\Delta S, n\Delta\tau)$$

Applying the Crank-Nicolson scheme, central differences for spatial derivatives can be used and an average of explicit and implicit time steps: $\frac{V_j^{n+1} - V_j^n}{\Delta\tau} = \frac{1}{2}\left[LV_j^{n+1} + LV_j^n\right]$,

where L is the spatial differential operator: $LV_j = \frac{1}{2}\sigma^2 S_j^2 \frac{V_{j+1} - 2V_j + V_{j-1}}{\Delta S^2} + rS_j \frac{V_{j+1} - V_{j-1}}{2\Delta S} - rV_j$

Rewriting in matrix form: $AV^{n+1} = BV^n$, where A and B are tridiagonal matrices.

The LU factorization method solves the linear system $Ax = b$ by decomposing A into: $A = LU$, where L is a lower triangular matrix and U is an upper triangular matrix.

Steps:

(a) decompose A into L and U

(b) solve $Ly = BV^n$ using forward substitution

(c) solve $UV^{n+1} = y$ using backward substitution

For a tridiagonal matrix, this method can be efficiently implemented in $O(N)$ operations. The Gauss-Seidel method iteratively refines the solution for $AV^{n+1} = BV^n$.

Starting with an initial guess $V^{n+1,(0)}$, the iterations update each component:

$$V_j^{n+1,(k+1)} = \frac{1}{A_{jj}}\left(B_j - \sum_{i<j} A_{ji} V_i^{n+1,(k+1)} - \sum_{i>j} A_{ji} V_i^{n+1,(k)}\right)$$

Iteration continues until convergence:

$$\|V^{n+1,(k+1)} - V^{n+1,(k)}\| < \epsilon$$

5. The following Python program implements the Crank-Nicolson finite difference method for solving the Black-Scholes PDE for a European call option.

```python
import numpy as np
import scipy.linalg
import scipy.stats as si
import matplotlib.pyplot as plt

def black_scholes(S, E, T, t, r, sigma):
    d1 = (np.log(S / E) + (r + 0.5 * sigma**2) * (T - t)) / (sigma * np.sqrt(T - t))
    d2 = d1 - sigma * np.sqrt(T - t)
    return S * si.norm.cdf(d1) - E * np.exp(-r * (T - t)) * si.norm.cdf(d2)

def crank_nicolson_european_call(S_max, E, T, r, sigma, M, N):
    """ Solves the Black-Scholes PDE for a European call using the Crank-Nicolson scheme """

    dS = S_max / M
    dt = T / N

    S = np.linspace(0, S_max, M+1)
    V = np.maximum(S - E, 0)

    alpha = 0.25 * dt * (sigma**2 * (np.arange(1, M) ** 2) - r * np.arange(1, M))
    beta = -0.5 * dt * (sigma**2 * (np.arange(1, M) ** 2) + r)
    gamma = 0.25 * dt * (sigma**2 * (np.arange(1, M) ** 2) + r * np.arange(1, M))

    A = np.diag(1 - beta) + np.diag(-alpha[1:], -1) + np.diag(-gamma[:-1], 1)
    B = np.diag(1 + beta) + np.diag(alpha[1:], -1) + np.diag(gamma[:-1], 1)

    A_inv = np.linalg.inv(A)

    for _ in range(N):
        V[1:M] = A_inv @ (B @ V[1:M])

    return S, V

S_max = 200
E = 100
T = 1
r = 0.05
sigma = 0.2

mesh_sizes = [50, 100, 200, 400]
exact_values = black_scholes(100, E, T, 0, r, sigma)

for M in mesh_sizes:
    N = M
    S, V = crank_nicolson_european_call(S_max, E, T, r, sigma, M, N)

    idx = np.argmin(np.abs(S - 100))
    numerical_value = V[idx]

```

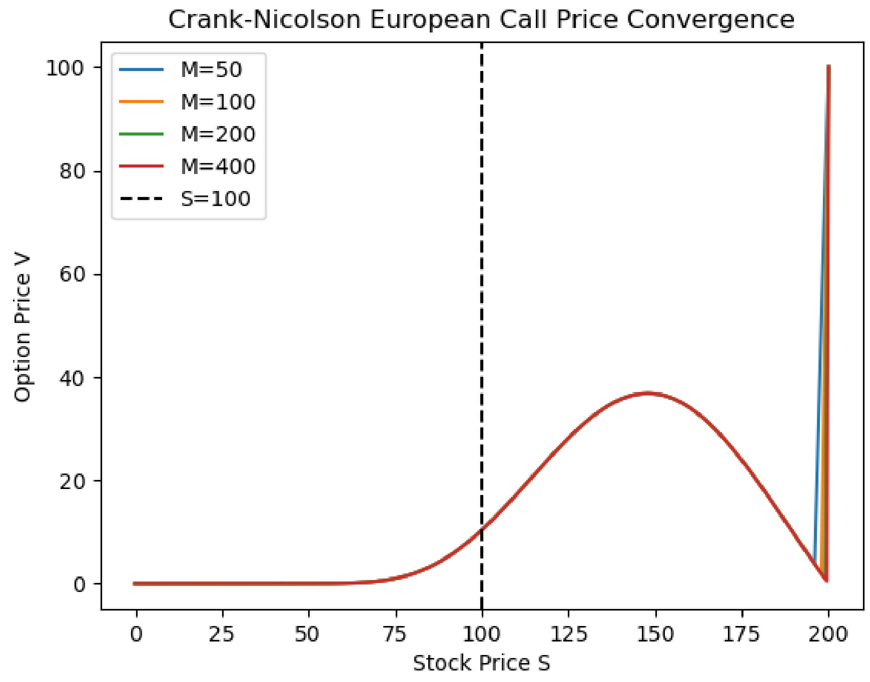
```
        print(f"Mesh size: {M}, Numerical Value: {numerical_value:.5f}, Exact 
            Value: {exact_values:.5f}, Error: {abs(numerical_value - 
            exact_values):.5f}")

        plt.plot(S, V, label=f'M={M}')

    plt.axvline(x=100, color='k', linestyle='--', label="S=100")
    plt.xlabel("Stock Price S")
    plt.ylabel("Option Price V")
    plt.title("Crank-Nicolson European Call Price Convergence")
    plt.legend()
    plt.show()
```

Output:

- Mesh size: 50, Numerical Value: 10.31711, Exact Value: 10.45058, Error: 0.13347
- Mesh size: 100, Numerical Value: 10.35367, Exact Value: 10.45058, Error: 0.09691
- Mesh size: 200, Numerical Value: 10.36277, Exact Value: 10.45058, Error: 0.08782
- Mesh size: 400, Numerical Value: 10.36504, Exact Value: 10.45058, Error: 0.08555



- The Crank-Nicolson method is effective for solving the Black-Scholes PDE and converges to the true value as the grid is refined
- The small remaining error is due to numerical discretization, which can be further reduced with higher-order schemes or adaptive meshing
- For practical pricing, using a sufficiently fine mesh (e.g. $M = 400$) provides a good balance between accuracy and computational cost