IEOR 221
Homework 8
Module 9 Options - Numerical Techniques

Louis Sallé-Tourne

November 8, 2024

Problem 2

The American call option price and the Greeks are calculated using a 100 step CRR binomial tree. The results are as follows:

- **American Call Option Price:** 13.2413
- **Delta:** 0.5481
- **Gamma:** 0.0112
- **Theta:** -6.0564
- **Vega:** 37.4385
- **Rho:** 33.9993

Formulas Used

- **Delta:**

$$\Delta \approx \frac{f_1^{t_1} - f_0^{t_1}}{S_1^{t_1} - S_0^{t_1}}$$

- **Gamma:**

$$\Gamma \approx \frac{\Delta_2 - \Delta_1}{S_1^{t_1} - S_0^{t_1}}$$

- **Theta:**
$$\Theta \approx \frac{f_1^{t_2} - f_0^{t_0}}{2\delta t}$$
- **Vega:**
$$\nu \approx \frac{f(\sigma + \delta\sigma) - f(\sigma)}{\delta\sigma}$$
- **Rho:**
$$\rho \approx \frac{f(r + \delta r) - f(r)}{\delta r}$$

Python Code:

```python
import numpy as np

# T : time to maturity
# r : risk-free rate
# q : dividend yield
# sigma : volatility
# N : number of steps in the binomial tree

def calculate_crr_parameters(T, r, q, sigma, N):
    """Calculate CRR model parameters: up factor (u), down
       factor (d), risk-neutral probability (p)."""
    dt = T / N
    discount = np.exp(-r * dt)
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    p = (np.exp((r - q) * dt) - d) / (u - d)
    return u, d, p, discount, dt


def build_stock_tree(S0, u, d, N):
    """Build the stock price tree."""
    stock_tree = np.zeros((N + 1, N + 1))
    for i in range(N + 1):
        for j in range(i + 1):
            stock_tree[j, i] = S0 * (u ** (i - j)) * (d**j)
    return stock_tree


def calculate_american_option_price(stock_tree, p, discount,
   K, N):
    """Calculate the American call option price using
       backward induction."""
```

```python
    option_tree = np.zeros((N + 1, N + 1))
    for j in range(N + 1):
        option_tree[j, N] = max(stock_tree[j, N] - K, 0)  # Payoff at maturity

    # Backward induction
    for i in range(N - 1, -1, -1):
        for j in range(i + 1):
            hold = discount * (
                p * option_tree[j, i + 1] + (1 - p) * option_tree[j + 1, i + 1]
            )
            exercise = stock_tree[j, i] - K
            option_tree[j, i] = max(hold, exercise)

    return option_tree[0, 0], option_tree


def calculate_delta(option_tree, stock_tree):
    """Calculate Delta using the difference in option prices at the first step."""
    delta = (option_tree[0, 1] - option_tree[1, 1]) / (
        stock_tree[0, 1] - stock_tree[1, 1]
    )
    return delta


def calculate_gamma(option_tree, stock_tree):
    """Calculate Gamma using central differences."""
    up = (option_tree[0, 2] - option_tree[1, 2]) / (stock_tree[0, 2] - stock_tree[1, 2])
    down = (option_tree[1, 2] - option_tree[2, 2]) / (
        stock_tree[1, 2] - stock_tree[2, 2]
    )
    gamma = (up - down) / (stock_tree[0, 1] - stock_tree[1, 1])
    return gamma


def calculate_theta(option_tree, dt):
    """Calculate Theta using finite difference."""
    theta = (option_tree[1, 2] - option_tree[0, 0]) / (2 * dt)
    return theta

```

```python

def calculate_vega(S0, K, T, r, q, sigma, N, american_call_price):
    """Calculate Vega by recalculating option price with bumped volatility."""
    bump_sigma = 0.01
    u_bump, d_bump, p_bump, discount_bump, _ = calculate_crr_parameters(
        T, r, q, sigma + bump_sigma, N
    )
    stock_tree_bump = build_stock_tree(S0, u_bump, d_bump, N)
    option_price_bump, _ = calculate_american_option_price(
        stock_tree_bump, p_bump, discount_bump, K, N
    )
    vega = (option_price_bump - american_call_price) / bump_sigma
    return vega


def calculate_rho(S0, K, T, r, q, sigma, N, american_call_price):
    """Calculate Rho by recalculating option price with bumped risk-free rate."""
    bump_r = 0.001
    u, d, p, discount_bump, _ = calculate_crr_parameters(
        T, r + bump_r, q, sigma, N
    )
    stock_tree = build_stock_tree(S0, u, d, N)
    option_tree_bump = np.zeros((N + 1, N + 1))

    # Rebuilding option tree with new discount factor
    for j in range(N + 1):
        option_tree_bump[j, N] = max(stock_tree[j, N] - K, 0)
    for i in range(N - 1, -1, -1):
        for j in range(i + 1):
            hold = discount_bump * (
                p * option_tree_bump[j, i + 1]
                + (1 - p) * option_tree_bump[j + 1, i + 1]
            )
            exercise = stock_tree[j, i] - K
            option_tree_bump[j, i] = max(hold, exercise)

    rho = (option_tree_bump[0, 0] - american_call_price) / bump_r
    return rho
```

```python


def calculate_all(S0, K, T, r, q, sigma, N):
    u, d, p, discount, dt = calculate_crr_parameters(T, r, q, sigma, N)
    stock_tree = build_stock_tree(S0, u, d, N)
    american_call_price, option_tree = calculate_american_option_price(
        stock_tree, p, discount, K, N
    )
    delta = calculate_delta(option_tree, stock_tree)
    gamma = calculate_gamma(option_tree, stock_tree)
    theta = calculate_theta(option_tree, dt)
    vega = calculate_vega(S0, K, T, r, q, sigma, N, american_call_price)
    rho = calculate_rho(S0, K, T, r, q, sigma, N, american_call_price)

    return american_call_price, delta, gamma, theta, vega, rho


# Define the parameters
S0 = 100.0  # initial stock price
K = 100.0  # strike price
T = 1.0  # time to maturity
r = 0.06  # risk-free rate
q = 0.06  # dividend yield
sigma = 0.35  # volatility
N = 100  # number of steps in the binomial tree

# Main calculations
american_call_price, delta, gamma, theta, vega, rho = calculate_all(
    S0, K, T, r, q, sigma, N
)

# Output results
print(f"American Call Option Price: {american_call_price:.4f}")
print(f"Delta: {delta:.4f}")
print(f"Gamma: {gamma:.4f}")
print(f"Theta: {theta:.4f}")
print(f"Vega: {vega:.4f}")
print(f"Rho: {rho:.4f}")
```

Listing 1: CRR Binomial Tree for American Call Option

Output:

```
American Call Option Price: 13.2413
Delta: 0.5481
Gamma: 0.0112
Theta: -6.0564
Vega: 37.4385
Rho: 33.9993
```