Untitled31

February 25, 2026

[11]:
```python
import numpy as np
import pandas as pd

def price_american_binomial(S0, K, T, r, sigma, N, option_type='call'):
    """Prices American options using a binomial tree with N steps."""
    dt = T / N
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    p = (np.exp(r * dt) - d) / (u - d)
    df = np.exp(-r * dt)

    # Stock prices at maturity
    S = S0 * d**(np.arange(N, -1, -1)) * u**(np.arange(0, N + 1, 1))

    # Terminal payoffs
    V = np.maximum(S - K, 0) if option_type == 'call' else np.maximum(K - S, 0)

    # Backward induction with early exercise check
    for i in range(N - 1, -1, -1):
        S = S0 * d**(np.arange(i, -1, -1)) * u**(np.arange(0, i + 1, 1))
        V_cont = df * (p * V[1:] + (1 - p) * V[:-1])
        # Check for early exercise
        intrinsic = np.maximum(S - K, 0) if option_type == 'call' else np..
↪maximum(K - S, 0)
        V = np.maximum(intrinsic, V_cont)

    return V[0]

def calculate_american_delta(S0, K, T, r, sigma, N, option_type):
    """Calculates Delta at t=0 for American options."""
    dt = T / N
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u

    # Option value at step 1
    v_up = price_american_binomial(S0 * u, K, T - dt, r, sigma, N - 1,␣
↪option_type)
```

```python
    v_down = price_american_binomial(S0 * d, K, T - dt, r, sigma, N - 1,␣
↪option_type)

    return (v_up - v_down) / (S0 * u - S0 * d)

# Parameters
params = {'S0': 100, 'K': 100, 'T': 0.25, 'r': 0.05, 'sigma': 0.20, 'N': 300}

# Q8: Pricing
am_call_price = price_american_binomial(**params, option_type='call')
am_put_price = price_american_binomial(**params, option_type='put')

# Q9: Delta
am_call_delta = calculate_american_delta(**params, option_type='call')
am_put_delta = calculate_american_delta(**params, option_type='put')

# Q10: Vega (Proxy: Change in price for 5% increase in sigma) [cite: 38]
am_call_vol_up = price_american_binomial(100, 100, 0.25, 0.05, 0.25, 300,␣
↪'call')
am_put_vol_up = price_american_binomial(100, 100, 0.25, 0.05, 0.25, 300, 'put')

print(f"American Call: Price={am_call_price:.2f}, Delta={am_call_delta:.4f}")
print(f"American Put:  Price={am_put_price:.2f}, Delta={am_put_delta:.4f}")
```

```
American Call: Price=4.61, Delta=0.5694
American Put:  Price=3.48, Delta=-0.4496
```

[13]:

```python
#Question 10

# Baseline Parameters
S0, K, T, r, N = 100, 100, 0.25, 0.05, 300
sigma_low = 0.20
sigma_high = 0.25

# Calculation
call_v20 = price_american_binomial(S0, K, T, r, sigma_low, N, 'call')
call_v25 = price_american_binomial(S0, K, T, r, sigma_high, N, 'call')
put_v20 = price_american_binomial(S0, K, T, r, sigma_low, N, 'put')
put_v25 = price_american_binomial(S0, K, T, r, sigma_high, N, 'put')

print(f"American Call Price (sigma=25%): {call_v25:.2f} (Increase of␣
↪{call_v25-call_v20:.2f})")
print(f"American Put Price (sigma=25%): {put_v25:.2f} (Increase of␣
↪{put_v25-put_v20:.2f})")
```

```
American Call Price (sigma=25%): 5.59 (Increase of 0.98)
American Put Price (sigma=25%): 4.46 (Increase of 0.98)
```

[16]:
```python
#Question 15 and 16
def price_trinomial(S0, K, T, r, sigma, N, option_type='call',␣
↪style='European'):
    """Trinomial tree pricing for European and American options."""
    dt = T / N
    dx = sigma * np.sqrt(3 * dt)

    # Probabilities (Boyle, 1986)
    pu = 0.5 * ((sigma**2 * dt + (r - 0.5 * sigma**2)**2 * dt**2) / dx**2 + (r␣
↪- 0.5 * sigma**2) * dt / dx)
    pd = 0.5 * ((sigma**2 * dt + (r - 0.5 * sigma**2)**2 * dt**2) / dx**2 - (r␣
↪- 0.5 * sigma**2) * dt / dx)
    pm = 1 - pu - pd
    df = np.exp(-r * dt)

    S = S0 * np.exp(dx * np.arange(-N, N + 1))
    V = np.maximum(S - K, 0) if option_type == 'call' else np.maximum(K - S, 0)

    for i in range(N - 1, -1, -1):
        V_cont = df * (pu * V[2:] + pm * V[1:-1] + pd * V[:-2])
        if style == 'American':
            S_node = S0 * np.exp(dx * np.arange(-i, i + 1))
            intrinsic = np.maximum(S_node - K, 0) if option_type == 'call' else␣
↪np.maximum(K - S_node, 0)
            V = np.maximum(intrinsic, V_cont)
        else:
            V = V_cont

    return V[0]

strikes = [90, 95, 100, 105, 110]
results = []

for K in strikes:
    results.append({
        'Strike': K,
        'Eur Call': price_trinomial(100, K, 0.25, 0.05, 0.2, 300, 'call',␣
↪'European'),
        'Eur Put': price_trinomial(100, K, 0.25, 0.05, 0.2, 300, 'put',␣
↪'European'),
    })

df_results = pd.DataFrame(results).round(2)
print(df_results)
```

```
   Strike  Eur Call  Eur Put
0      90     11.67     0.55
```

```
1     95     7.71     1.53
2    100     4.61     3.37
3    105     2.48     6.17
4    110     1.19     9.83
```

[ ]: