theta-and-gamma

January 7, 2024

[14]:
```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np
```

[15]:
```
def black_scholes_call_price(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price

def calculate_theta(S, K, T, r, sigma, option_type='call'):
    epsilon = 1e-4  # a small time increment for numerical differentiation
    t1 = T - epsilon
    t2 = T

    if option_type == 'call':
        price_t1 = black_scholes_call_price(S, K, t1, r, sigma)
        price_t2 = black_scholes_call_price(S, K, t2, r, sigma)
    else:
        # Add logic for put option if needed
        pass

    theta = (price_t2 - price_t1) / epsilon
    return theta

def calculate_gamma(S, K, T, r, sigma, option_type='call'):
    epsilon = 1e-4  # a small price increment for numerical differentiation
    S1 = S - epsilon
    S2 = S + epsilon

    if option_type == 'call':
        price_S1 = black_scholes_call_price(S1, K, T, r, sigma)
        price_S2 = black_scholes_call_price(S2, K, T, r, sigma)
        gamma = (price_S2 - 2 * black_scholes_call_price(S, K, T, r, sigma) +␣
 ↪price_S1) / epsilon**2
```

```
    else:
        # Add logic for put option if needed
        pass

    return gamma
```

[20]:
```
# Example usage:
ticker = 'AAPL'
option_strike = 150
option_expiry_days = 30
risk_free_rate = 0.02
volatility = 0.3

# Fetch historical data for the underlying asset (e.g., stock)
stock_data = yf.download(ticker, start='2023-01-01', end='2024-01-31')
```

```
[**********************100%%**********************]  1 of 1 completed
```

[25]:
```
stock_data.iloc[-1] /100
```

[25]:
```
Open              1.8199
High              1.8276
Low               1.8017
Close             1.8118
Adj Close         1.8118
Volume       623033.0000
Name: 2024-01-05 00:00:00, dtype: float64
```

[22]:
```
# Ensure that the data is not empty
if stock_data.empty or len(stock_data) < 2:
    print("Insufficient data. Please check the date range.")
else:
    # Assuming the current stock price is the last closing price
    current_stock_price = stock_data['Close'].iloc[-1]

    # Convert days to years for time to expiration
    time_to_expiration = option_expiry_days / 365.0

    # Calculate theta and gamma
    theta_value = calculate_theta(current_stock_price, option_strike,␣
 ↪time_to_expiration, risk_free_rate, volatility)
    gamma_value = calculate_gamma(current_stock_price, option_strike,␣
 ↪time_to_expiration, risk_free_rate, volatility)

    print(f'Theta: {theta_value:.4f}')
    print(f'Gamma: {gamma_value:.4f}')
```

```
Theta: 5.9025
Gamma: 0.0020
```

[24]:
```
stock_data.std() / 100
```

[24]:
```
Open              0.174535
High              0.171833
Low               0.174522
Close             0.172473
Adj Close         0.174220
Volume       177121.300465
dtype: float64
```

[ ]:
```

```