derivatives-pricing

October 1, 2023

[1]:
```python
import numpy as np
from scipy.stats import norm
import yfinance as yf
import matplotlib.pyplot as plt
```

[2]:
```python
# Define the option parameters
S0 = 100          # Current stock price
T = 1.0           # Time to expiration (in years)
r = 0.05          # Risk-free interest rate
sigma = 0.2       # Volatility (annualized)
```

[3]:
```python
# Fetch historical stock price data
ticker_symbol = "AAPL"  # Replace with the symbol of the stock you want to use
stock_data = yf.Ticker(ticker_symbol)
historical_data = stock_data.history(period="1y")
```

[4]:
```python
# Calculate daily returns
historical_data['Daily Returns'] = historical_data['Close'].pct_change().
 ↪dropna()
```

[5]:
```python
# Calculate annualized volatility from daily returns
annualized_volatility = historical_data['Daily Returns'].std() * np.sqrt(252)
```

[6]:
```python
# Black-Scholes-Merton formula for European call option
def black_scholes_call(S0, K, T, r, sigma):
    d1 = (np.log(S0 / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S0 * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price
```

[7]:
```python
# Create an array of strike prices
strike_prices = np.linspace(80, 120, 41)  # Vary strike prices from 80 to 120
```

[8]:
```python
# Calculate call option prices for different strike prices
call_option_prices = [black_scholes_call(S0, K, T, r, annualized_volatility)␣
 ↪for K in strike_prices]
```

[9]:
```
# Create a plot
plt.figure(figsize=(10, 6))
plt.plot(strike_prices, call_option_prices, label="Call Option Price")
plt.xlabel("Strike Price")
plt.ylabel("Option Price")
plt.title("European Call Option Price vs. Strike Price")
plt.legend()
plt.grid(True)

# Show the plot
plt.show()
```