Problem 3

Friday, November 1, 2024 12:46 PM

a) The put call parity,

$$c + Ke^{-rt} = p + Se^{-qt}$$

$$\Rightarrow q = \frac{1}{t} \ln\left(\frac{c - p + Ke^{-rt}}{S}\right)$$

Thus,

$$q = -\frac{1}{0{,}25} \cdot \ln\left(\frac{78 - 26 + 950 \cdot e^{-0{,}04 \cdot 0{,}25}}{1000}\right) = 0{,}0299$$

Thus, the dividend yield is 2,99%

b using black Scholes for european call option

$$c = Se^{-qt} N(d_1) - Ke^{-rT} N(d_2)$$

$$\Rightarrow d_1 = \frac{\ln\left(\frac{S}{K}\right) + (r - q + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

Thus

$$d_1 = \frac{\ln\left(\frac{1000}{950}\right) + (0{,}04 - 0{,}0299 + 0{,}5 \cdot 0{,}0625) \cdot 0{,}25}{0{,}125} = 0{,}4915$$

$$d_2 = 0{,}4915 - 0{,}125 = 0{,}3665$$

Thus the normal distribution,

$$N(d_1) = N(0{,}4915) \approx 0{,}6889$$

$$N(d_2) = N(0{,}3665) \approx 0{,}6446$$

$$c = Se^{-qt} N(d_1) - Ke^{-rt} N(d_2)$$

$$= 1000 \cdot e^{-0{,}0299 \cdot 0{,}25} \cdot 0{,}6889 + 950 e^{-0{,}04 \cdot 0{,}25} \cdot 0{,}6446 = 684{,}97 - 608{,}27 = 76{,}70$$

Since the theoretical is larger I need to use python, using Newton Raphson I get 24,68%.

Code is below,

```python
import math
from scipy.stats import norm
def black_scholes_call(S, K, T, r, q, sigma):
    """Calculate the Black-Scholes price of a European call option."""
    d1 = (math.log(S / K) + (r - q + 0.5 * sigma ** 2) * T) / (sigma * math.sqrt(T))
    d2 = d1 - sigma * math.sqrt(T)
```

```python
    call_price = S * math.exp(-q * T) * norm.cdf(d1) - K * math.exp(-r * T) * norm.cdf(d2)
    return call_price
def implied_volatility_call(S, K, T, r, q, market_price, tol=1e-6, max_iterations=100):
    """Calculate the implied volatility of a European call option using the bisection
method."""

    sigma_low = 1e-6
    sigma_high = 5.0
    for i in range(max_iterations):
        sigma_mid = (sigma_low + sigma_high) / 2.0
        price = black_scholes_call(S, K, T, r, q, sigma_mid)
        diff = price - market_price

        print(f"Iteration {i+1}: sigma = {sigma_mid:.6f}, price = {price:.6f}, diff =
{diff:.6f}")
        if abs(diff) < tol:
            return sigma_mid  # Implied volatility found
        elif diff > 0:
            sigma_high = sigma_mid  # Volatility too high
        else:
            sigma_low = sigma_mid  # Volatility too low
    # If convergence not reached, return the best estimate
    return sigma_mid
# Given parameters
S = 1000         # Spot price
K = 950          # Strike price
T = 0.25         # Time to expiration in years
r = 0.04         # Risk-free rate
q = 0.0299       # Dividend yield from part (a)
market_price = 78  # Market price of the call option
# Calculate the implied volatility
implied_vol = implied_volatility_call(S, K, T, r, q, market_price)
print(f"\nImplied Volatility: {implied_vol * 100:.2f}%")
```

Output:

```
Iteration 1: sigma = 2.500001, price = 478.743576, diff = 400.743576
Iteration 2: sigma = 1.250001, price = 263.985877, diff = 185.985877
Iteration 3: sigma = 0.625001, price = 147.774083, diff = 69.774083
Iteration 4: sigma = 0.312501, price = 89.718491, diff = 11.718491
Iteration 5: sigma = 0.156251, price = 62.990794, diff = -15.009206
Iteration 6: sigma = 0.234376, price = 75.838078, diff = -2.161922
.
.
.
Iteration 24: sigma = 0.246797, price = 77.999988, diff = -0.000012
Iteration 25: sigma = 0.246797, price = 78.000014, diff = 0.000014
Iteration 26: sigma = 0.246797, price = 78.000001, diff = 0.000001

Implied Volatility: 24.68%
```